SHORTEST PATH PROBLEM WITH DISJUNCTIVE CONFLICT CONSTRAINTS

by

Bahadır Pamuk

B.S., Industrial Engineering, Boğaziçi University, 2016

M.S., Industrial Engineering, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

I am proud to acknowledge that this dissertation is solely guided by pure scientific curiosity and without any financial support, except my regular salary to a certain extent as a TA. To that end, I am deeply grateful to my dearest wife Begüm Göloğlu Pamuk. Without her unconditional emotional support, this thesis would not be completed.

I am profoundly grateful to my advisor, Prof. İ. Kuban Altınel, for his sage advice, rigorous academic guidance, and endless appetite for academic freedom and autonomy. His expertise in combinatorial optimization was indispensable for this work.

I am grateful to my co-advisor, Prof. Temel Öncan, for his guidance in experimentation, which is completed on servers provided by Galatasaray University.

I am deeply thankful to my mentor Prof. Ümit Bilge. She is not only a jury member for this dissertion, but the coordinator for BUFAIM, my beloved lab. Her guidance and perfectionism deepened my analytical skills and my attention to detail.

I am also deeply thankful for my family, especially my mother Aycan Pamuk. Her continuous support and belief in me have been instrumental in my success. I must not forget dear BUFAIM members, Gökalp and Mert, I learned so much from you.

I would like to thank Prof. Caner Taşkın for his consistent support and guidance during my master's dissertation and also as a progress jury member for this work. I would like to express my thanks to Prof. Selda Küçükçifçi for her time and contributions as a continuous jury member from the start. I would like to thank Prof. Taner Bilgiç, and Assist. Prof. Ezgi Karabulut Türkseven for their time as a jury member.

At last but not the least, I would like to thank all faculty members, secretaries, TAs, and workers at the Industrial Engineering Department of Boğaziçi University.

ABSTRACT

SHORTEST PATH PROBLEM WITH DISJUNCTIVE CONFLICT CONSTRAINTS

An extension of the ordinary one-to-many shortest path problem that also considers additional disjunctive conflict relations between the arcs is considered in this study. In this problem called SPC, optimal shortest path tree is not allowed to include any conflicting arc pair. As is the case with many polynomially solvable combinatorial optimization problems, the addition of conflict relations makes the problem $\mathcal{NP}$-hard. We propose three novel algorithms for SPC. One such algorithm is a novel branch-and-bound algorithm, which benefits from the solution of the one-to-many shortest path relaxation, an efficient primal-dual re-optimization scheme and a fast infeasibility detection procedure for pruning the branch-and-bound tree. The second solution method is a greedy depth first search algorithm, which benefits from the sufficient conditions of infeasibility special to shortest path problem. The third one is also a branch-and-bound algorithm; but it searches through the solution space using the stable sets of the conflict graph, benefiting from an efficient circuit prevention and a fast infeasibility detection procedure for pruning the branch-and-bound tree. Several MILP formulations are tested in order to find the best one. At last, new algorithms are extensively tested on both randomly generated and realistic test instances, comparing them with the best MILP formulation implemented with a state-of-the-art commercially available MILP solver. According to the obtained results, it is possible to say that the best of the novel algorithms outperforms the solver, while the other two's performances are comparable.

ÖZET

AYIRICI ÇATIŞMA KISITLI EN KISA YOL PROBLEMİ

Bu çalışmada oklar arasındaki ayrık çatışma ilişkilerini de dikkate alan birden çoka en kısa yol probleminin bir uzantısı ele alınmaktadır. Ayırıcı çatışma kısıtlı en kısa yol problemi olarak adlandırılan bu problemde, en kısa yol ağacının herhangi bir çatışan ok çifti içermesine izin verilmez. Polinom olarak çözülebilen birçok birleşimsel eniyileme probleminde olduğu gibi, çatışma ilişkilerinin eklenmesi problemi $\mathcal{NP}$-zor yapar. Bu tez kapsamında üç yeni algoritma önerilmektedir. Önerilen ilk algoritma, birden çoka en kısa yol probleminin çözümünden, etkili bir güncelleme yordamından ve dallanmayı budamak için hızlı bir olursuzluk saptama yönteminden yararlanan bir dal ve sınır algoritmasıdır. İkinci olarak, en kısa yol problemine özgü olurluluk gerekli koşullarından yararlanan, derinlik öncelikli bir dalış algoritması geliştirilmiştir. Üçüncü olarak, çatışma çizgesinin bağımsız kümelerini kullanan, verimli bir tur önleyici dizgeden ve yine hızlı bir olursuzluk saptayıcısından yararlanan bir dal ve sınır algoritması geliştirilmiştir. Son olarak, hem rastgele oluşturulmuş hem de gerçekçi test örnekleri üzerinde kapsamlı bir şekilde test edilen yeni algoritmalar, piyasada bulunan en son teknolojiye sahip bir karma tamsayılı doğrusal program çözücü ile karşılaştırılmıştır. Kapsamlı bilgisayısal deneylere göre, yeni algoritmalardan en etkin olanı çözücüden çok daha başarılıdır. Diğerlerinin başarımları ise karşılaştırılabilir düzeydedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$A(N)$	Arcs of Network N
c_{ij}	Cost of Arc (i, j)
$\overline{c}_{ij}$	Reduced Cost of Arc (i, j)
C	Conflict Graph
$C^{(t)}$	The Conflict Graph of Sub-problem t
$\widehat{C}$	Extended Conflict Graph
$Cost(\theta)$	Objective Value of Problem θ
$d_G(v)$	Degree of Vertex v in Graph G
$d_N^-(v)$	Indegree of Vertex v in Network N
$d_N^+(v)$	Outdegree of Vertex v in Network N
$f(n)$	Any Polynomial Time Computable Function
f_a	Decision Variable Representing the Amount of Flow on Arc a
$F^{(t)}$	The Set of Free Arcs of Sub-problem t
$G[K]$	Sub-graph of G Induced by Vertex Subset $K \subseteq V(G)$
$\overline{G}$	The Complement of Graph G
$I^{(t)}$	The Set of Included Arcs of Sub-problem t
$\mathcal{K}(C)$	The Family of Maximal Cliques of C
$\mathcal{L}$	Active Sub-problem List for an Algorithm
N	Network
$N^{(t)}$	Network of Sub-problem t
$N_C(a)$	The Set of Arcs Neighboring a on C
$\overline{N}_C(a)$	The Set of Arcs Not Neighboring a on C
O	A Non-decreasing Degree Ordering of Arcs
$\mathcal{O}$	Worst Case Complexity Order
$\mathcal{P}_F$	Polyhedron of Feasible Solutions of LP Relaxation of F
Q_r	Candidate Arcs in Iteration r of Primal-dual Algorithm
s	Source Vertex for SP
$(S, \overline{S})$	The Set of Arcs from S to $\overline{S}$

S^{new}	New Vertices Included in Tree after a Primal-dual Iteration
$SP^{(t)}$	The Relaxed SP of Sub-problem t
$SPC^{(t)}$	Sub-problem t
$\mathscr{S}$	A Stable Set
T	A Tree of Graph
T^*	A Directed Spanning Outtree with Minimum Total Distance
$T^{(t)}$	SPT of Sub-problem t
$V(N)$	Vertices of Network N
W	A Vertex Maximal Subset of $V(C)$
x_a	Binary Decision Variable for Selecting Arc a
$X^{(t)}$	The Set of Excluded Arcs of Sub-problem t
$\bar{z}$	The Objective Value of Incumbent Solution
$\alpha(G)$	Stability number of Graph G
$\gamma_G(W)$	The Set of Edges with Both Endpoints in W
Γ	Network Density
$\delta_G(W)$	The Set of Edges with One Endpoint in W
ϵ	Amount of Decrease in the Number of Free Arcs
θ	A Parent Problem in an Algorithm
ξ	Generated Sub-problem After Branching in an Algorithm
π_v	Dual Variables Associated with Vertex v
Σ	The Family of Conflict Free Spanning Outtrees of N
τ	An Upper Bound on the Performance Ratio
$\omega(G)$	The Clique Number of Graph G
$\omega_c(G)$	The Weighted Clique Number of Graph G
Ω	Conflict Density

LIST OF ACRONYMS/ABBREVIATIONS

ATSP	Asymmetric Traveling Salesman Problem
BB	Branch and Bound Algorithm
BBA	Branch and Bound Algorithm Based on Conflicts
BBSS	Branch and Bound Algorithm Based on Stable Sets
BFS	Breadth First Search Algorithm
FPCP	Forbidden Pairs Constrained Path Problem
FTCP	Forbidden Transition Constrained Path Problem
LB	Lower Bound
LP	Linear Programming
MC	Maximum Clique Problem
MCF	Minimum Cost Flow Problem
MCFC	Minimum Cost Flow Problem with Conflicts
MILP	Mixed Integer Linear Programming
MS	Maximum Cardinality Stable Set Problem
MWC	Maximum Weight Clique Problem
MWS	Maximum Weight Stable Set Problem
PECP	Properly Edge Colored Path Problem
RT-PCR	Reverse Transcription Polymerase Chain Reaction
SLB	Smallest Lower Bound
SOS 1	Special Ordered Set Type 1
SP	One-to-many Shortest Path Problem
SPC	One-to-many Shortest Path Problem with Conflicts
SPT	Shortest Path Tree
TSPLIB	A Library of Test Instances for Traveling Salesman Problem
UB	Upper Bound

1. INTRODUCTION

Combinatorial optimization problems with conflicts have recently become attractive research topics as a consequence of their potential applicability in real-life. From a general perspective, their feasible solution sets consist of the intersection of the feasible solutions of a well-known classical optimization problem, e.g., one-to-many shortest path problem, with the stable sets expressed by means of the additional conflict constraints. In short, any feasible solution is essentially a stable set of a simple graph representing the conflict relations, i.e., *conflict graph*, which has the additional property characterized by the constraints of the classical problem. As for example, we can give the transportation problem with conflict constraints [1–3], the minimum spanning tree problem with conflict constraints [4–9], the assignment problem with conflict constraints [10–12], the maximum weight perfect matching problem with conflict constraints [13], the maximum flow problem with conflict constraints [14, 15], and the minimum cost flow problem with conflict constraints [16, 17].

In this work, we are particularly concerned with the one-to-many *Shortest Path Problem with Conflicts* (SPC), which is a recent extension of the one-to-many *Shortest Path Problem* (SP). Due to its importance in Network Optimization, many efficient algorithms have been developed for SP. It has not only numerous applications but also emerges as a sub-problem while solving many difficult combinatorial optimization problems [18]. Recall that, SP seeks all paths having minimum lengths from a particular source vertex to every other one, on a given network with known arc weights. In fact, an optimal solution is a spanning outtree (i.e., a directed spanning tree with arcs pointing away from the source vertex), which is known as the *Shortest Path Tree* (SPT). SPC includes conflict constraints in addition, and involves in the determination of a conflict free SPT. Conflict constraints prohibit the simultaneous inclusion of some pairs of conflicting arcs in the SPT. These side constraints are also qualified as disjunctive or exclusionary, and can be considered within the context of forbidden arc pairs. They

are represented as SOS 1 type constraints [19] as well. It is known that their addition to SP makes SPC $\mathcal{NP}$-hard [20].

SP is a special case of the *Minimum Cost Flow Problem* (MCF) with a single source having a supply of one less than the number of vertices. As a consequence, SPC is a special case of the *Minimum Cost Flow Problem with Conflicts* (MCFC). Best known algorithms for MCF includes augmenting path algorithm which involves successive solution of SP [18]. Therefore, an efficient method for SPC could possibly provide us with an efficient method for MCFC or other network flow problems with conflicts by transforming the original problem into a sequence of shortest path problems with conflicts.

SP has many real world applications in many fields: transportation, telecommunication, facility layout optimization, robotics, path finding, etc. all require some form of efficient SP algorithms. Therefore an efficient SPC algorithm can provide us a tool to deal with SOS 1 type constrained versions of these real world problems.

In the remaining part of this thesis, we first provide in Chapter 2 basic concepts and definitions that can increase the readability. Chapter 3 consists of a critical summary of the related work and our motivation. Formulations of SPC are provided in Chapter 4. Reoptimization of SP is discussed in detail in Chapter 5. The proposed novel *Branch-and-Bound* algorithm (BB), including the new primal-dual reoptimization scheme, is explained in detail in Chapter 6. A diving algorithm is introduced in Chapter 7. Another BB based on stable sets of the conflict graph is presented in Chapter 8. The results of extensive computational tests are reported in Chapter 9. Finally, Chapter 10 includes conclusions and potential future research directions. The reader can find the detailed lists of random and realistic test instances in Appendix A, and detailed computational results in Appendix B.

2. BASIC CONCEPTS AND NOTATION

SPC is defined on a simple network (directed graph) $N = (V(N), A(N))$ having neither parallel arcs nor loops. $V(N)$ and $A(N)$ are respectively the sets of vertices and arcs. Besides, a non-negative weight c_a and a list $N_C(a)$ of arcs conflicting with a are associated with each arc $a \in A(N)$. Conflicts can be modeled by means of a conflict graph $C = (V(C), E(C))$, which is a simple (undirected) graph, where each arc in $A(N)$ is represented as a vertex in $V(C)$, i.e., $V(C) \equiv A(N)$, and each edge in $E(C)$ corresponds to a conflicting arc pair. We assume that conflict is a symmetric relation, i.e., for any arc pair a and b, a conflicts with b if and only if b conflicts with a. Hence, the unordered pair $\{a, b\}$ is an edge of C, i.e., $\{a, b\} \in E(C)$.

Let $G = (V(G), E(G))$ be a graph that is composed of vertex set $V(G)$ and edge set $E(G)$. For a subset of vertices $W \subseteq V(G)$, $\delta_G(W) \subseteq E(G)$ represents the set of edges with one endpoint in W, and $\gamma_G(W) \subseteq E(G)$ stands for the set of edges with both endpoints in W. Then, the degree of a vertex v of a graph G is $d_G(v) = |\delta_G(v)|$. $G[W]$ denotes the sub-graph of G induced by the vertex subset W. Hence, $G[W] = (W, \gamma_G(W))$, i.e., two vertices of $G[W]$ are adjacent if they are also adjacent in G. The set of all neighbours in graph G of a vertex v excluding itself is denoted as $N_G(v) = \{u : \{v, u\} \in E(G)\}$ and its non-neighbours are denoted as $\overline{N}_G(v) = \{u : \{v, u\} \notin E(G)\}$. Similarly, the set of neighbours of a vertex subset $W \subseteq V(G)$ is $N_G(W) = \bigcup_{v \in W} N_G(v)$. The ratio of the number of edges (arcs) over the maximum possible number of edges (arcs) yields the density of a graph (network).

A subset of vertices $K \subseteq V(G)$ is a *clique* of G if the induced sub-graph $G[K]$ is complete. Notice that this is a *hereditary* property, i.e., any sub-graph of a clique is also a clique. K is a maximal clique of G if it is not included properly within another one. *Maximum Clique Problem* (MC) determines a clique of maximum cardinality in graph G. This maximum cardinality is the *clique number* $\omega(G)$ of G. *Maximum Weight Clique Problem* (MWC) aims to determine a clique of maximum total weight in a graph

G where each vertex v has a non-negative weight c_v, and the *weighted clique number* $\omega_c(G)$ is the weight of a maximum weight clique. Both MC and MWC are $\mathcal{NP}$-hard problems [21].

A *stable set* (or in other words vertex packing or independent set) of a graph G is a subset of vertices $\mathcal{S} \subseteq V(G)$ such that no two vertices of $\mathcal{S}$ are adjacent. This is again a *hereditary* property and $\mathcal{S}$ becomes maximal if it is not included in another stable set properly. A stable set $\mathcal{S}$ is of maximum cardinality if there is no other stable set of size larger than $|\mathcal{S}|$ in G. The size of the maximum stable set, i.e., $\alpha(G)$, is referred to as the *stability number* of G. When we find all maximal cliques of G, we can solve the *Maximum Cardinality Stable Set Problem* (MS) for G by selecting one vertex from each maximal clique of G. When each vertex v of G has a non-negative weight c_v one can speak about the *Maximum Weight Stable Set Problem* (MWS), which aims to find a stable set of G with maximum total weight. This value is the *weighted stability number* of G, i.e., $\alpha_c(G)$. Both MS and MWS are $\mathcal{NP}$-hard problems [21]. A clique of graph G is a stable set of its complement $\overline{G}$. As a result we can solve MC and MWC on G by respectively solving MS and MWS on $\overline{G}$, and vice versa.

The *indegree* of a vertex v of a network N is denoted as $d_N^-(v)$ and it gives the number of arcs with head v (inarcs), in network N. Similarly, the *outdegree* of a vertex v is denoted as $d_N^+(v)$ and it gives the number of arcs with tail v (outarcs), in network N. Clearly, $d_N^-(v) = |\delta_N^-(v)|$ and $d_N^+(v) = |\delta_N^+(v)|$ with $\delta_N^-(v)$ and $\delta_N^+(v)$ denoting respectively the inarcs and outarcs of v.

A tree $T = (V(T), E(T))$ ($T = (V(T), A(T))$ of graph G (network N) is a subgraph (sub-network) of G (N) that is acyclic (circuit free) and connected (weakly connected). A spanning tree T of G (N) is a tree, which spans $V(G)$ (($V(N)$, i.e., $V(T) \equiv V(G)$ ($V(T) \equiv V(N)$). A directed spanning tree of N is an *outtree* or arborescence (*intree* or anti-arborescence) if all of its arcs point away from its root (point towards the root).

3. RELATED WORKS AND MOTIVATION

Ever since Krause et al.'s seminal work [22], a considerable amount of research on paths with conflicts has been accomplished. Some of the characteristic examples related to SPC under three groups, which are paths avoiding forbidden pairs, paths avoiding forbidden transitions, and properly edge colored paths are provided next.

3.1. Paths Avoiding Forbidden Pairs

SPC has its roots in *Forbidden Pairs Constrained Path problem* (FPCP): given a network $N = (V(N), A(N))$, a fixed source and target vertices $s, t \in V(N)$, and a set F of forbidden pairs, the aim is to determine a directed s,t-path that contains at most one element from each pair in F. The first appearance of FPCP is related to automatic software testing and validation. Execution flow in a computer program can be modeled as a network where s and t respectively denote the beginning and end of a run. In this setting vertices and arcs between them represent respectively basic blocks and sequence of their executions. Notice that this network is acyclic. Hence, a directed s,t-path is a test path and automatic generation of all test paths that cover basic blocks, which enables checking the program flow from every branching statement to every exit, is highly desirable. An important problem is a test path's being unexecutable due to the inclusion of mutually unexecutable statements. A reasonable approach for eliminating such paths is suggested in [22]. According to them, first mutually unexecutable statements of the programs can be determined through a careful semantic analysis; and then, test paths can be generated by allowing the inclusion of at most one statement of each unexecutable pair. In [22] each forbidden arc pair represents an unexecutable statement pair. However, in the succeeding work [23], forbidden vertex pairs are used with the same purpose and it is shown that the problem of building a directed path satisfying forbidden vertex pair restrictions is $\mathcal{NP}$-complete on acyclic networks. Another study considering FPCP [24] proposes an algorithm to generate a forbidden vertex pair constrained s,t-path in an execution flow network. The

complexity of the algorithm is polynomial in the number of vertices and the number of forbidden vertex pair constrained paths actually existing in the program, when the network is acyclic. In [25] the problem under skew symmetry conditions of the vertex pairs is studied: for each vertex pair $\{u_1, u_2\}$ and $\{v_1, v_2\}$ if (u_1, v_1) is an arc of the network, then (u_2, v_2) is also an arc. Then, FPCP can be solved in polynomial time. In particular, they show that this problem is polynomially equivalent to finding an augmenting path with respect to a given matching. In [26] additional complexity results is provided by considering two different structures of the forbidden vertex pairs. The first one is the halving structure where for any two forbidden pairs $\{u_1, v_1\}$ and $\{u_2, v_2\}$ there is neither a directed v_1, u_2-path nor $v_1 = u_2$. The second one is the hierarchical structure where no two pairs $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are on the same directed u_1, v_2-path in u_1, u_2, v_1, v_2 order. They show that for even acyclic networks having forbidden vertex pairs with the halving structure the determination of a directed forbidden vertex pair constrained s, t-path is $\mathcal{NP}$-hard and provide a simple polynomial time algorithm for the same problem with the hierarchical structure.

FPCP has applications in bio-informatics as well. The determination of the sequence of its amino acids from a set of masses corresponding to the masses of prefix and suffix fragments, namely a mass spectrum, of a peptide with unknown structure is an important task. Construction of an acyclic network from the mass spectrum using the information about the weights of peptides is suggested in [27]. They also derive forbidden vertex pairs from the mass spectrum. Forbidden pairs have both halving and hierarchical structure of [26], i.e., every two forbidden vertex pairs $\{u_1, v_1\}$ and $\{u_2, v_2\}$ satisfy either the topological order u_1, u_2, v_2, v_1 or u_2, u_1, v_1, v_2 on a forbidden pair constrained directed s, t-path. Then, a directed forbidden pair constrained s, t-path gives an amino acid sequence of the peptide, which can be done by a polynomial time dynamic programming algorithm [27].

FPCP on acyclic networks with forbidden vertex pairs also appears in gene finding using RT-PCR tests [28]. Non-overlapping segments of DNA sequences can be represented as the vertices with weights being the number of nucleotides in this segment,

and the precedence relations between the segments in some gene transcript as the arcs of a network. This is the so called splicing graph and paths correspond to predicted genes. Then, real genes can be identified by determining directed s,t-paths with the highest score in the given splicing graph using the information provided by a set of RT-PCR tests. A simpler version of this problem can be obtained by representing tests as pair of vertices and letting forbidden pairs be the inconsistent ones. Then, true gene identification with RT-PCR tests becomes finding a directed s,t-path allowing at most one of the vertices from each forbidden pairs. In [29] a detailed complexity analysis of this version is provided under different assumptions on the structure of the set of inconsistent tests, i.e., forbidden pairs.

Another application of FPCP belongs to aircraft routing, where each feasible route is a path in an airway network that must satisfy restrictions imposed by the air traffic service authorities. In [30] it is shown that, after some simplification these restrictions can be modelled by means of forbidden arc pairs and the determination of a feasible route becomes FPCP. A polyhedral study of this problem is realized in [31]. They introduce a new family of valid inequalities for FPCP polytope where forbidden pairs are arc pairs and show that they are sufficient to provide a complete linear description in the special case where the forbidden arc pairs are disjoint. Furthermore, they show that the number of facets of the FPCP polytope is exponential in the size of the graph, even for the case of a single forbidden arc pair. This work and the seminal paper [22] differ from all above mentioned ones in their preference of conflicting pairs from the set of arcs instead of vertices. In addition, arc weights are also considered and the problem is stated as the determination of an s,t-path avoiding the inclusion of both arcs of the forbidden pairs with minimum total weight; in fact this is a one-to-one shortest path problem with forbidden arc pairs. SPC is even harder than FPCP. It is strongly inapproximable even if the conflict relations can be represented by a 2-ladder as shown in [20], as a consequence of a related result on forbidden pairs of vertices [32].

Arc and vertex weights for the problem with forbidden vertex pairs are also considered in [26]. Their algorithm yields solutions also for several optimization versions

of the FPCP problem, such as the search for the shortest or the longest path with the length defined as the sum of arc or vertex weights, or the search for the k shortest or k longest paths. One-to-one shortest path problem with forbidden vertex pairs is studied in [33] and BB and dynamic programming exact solution algorithms are proposed. In their recent work, [34] revisit software testing and propose a two-stage matheuristic for an extension of the one-to-one shortest path problem with forbidden arc pairs. Their model allows the inclusion of the forbidden arc pairs in the solution with a penalty. A penalty also occurs when none of the arcs of a forbidden pair is in the solution. The aim is to determine an s,t-path having the smallest overall cost, which is defined as the sum of the traversal costs of the selected arcs and the penalties associated with the forbidden arc pairs on the path.

3.2. Paths Avoiding Forbidden Transitions

FPCP has a special form called *Forbidden Transition Constrained Path problem* (FTCP), where each pair forms a *transition* [35]. A transition in a graph (network) is a pair of adjacent edges (arcs). It is forbidden when at most one of the two adjacent edges (arcs) can be traversed. For example if a transition is forbidden then the corresponding adjacent edge (arc) pair cannot be on a feasible path (directed path). Then, FTCP deals with the determination of paths without forbidden transitions on a given graph with a list of such transitions. FTCP can be faced within many contexts. For example in optical networks nodes configured asymmetrically can be represented by vertices with forbidden transitions, which makes the routing problem in optical networks an application of FTCP [36–38]. Also, turns are restricted in order to streamline traffic flow at the intersections of the real life road networks. Each restriction can be modeled by a forbidden transition where the two adjacent arcs share the vertex representing the intersection, which relates FTCP to vehicle routing [39,40]. There is a rich literature on the computational complexity of FTCP [41]. Finding an elementary s,t-path avoiding forbidden transitions on graphs [42] and digraphs [43] are both $\mathcal{NP}$-complete problems, which is also true even on grids [44].

3.3. Properly Edge Colored Paths

In edge-colored graphs, which are introduced in [45] for studying chromosome arrangements, a pair of adjacent edges sharing a vertex form a forbidden transition if both have the same color. As a result a forbidden transition constrained path becomes equivalent to a *Properly Edge Colored Path* (PECP), namely to a path that does not use consecutively two edges having the same color when the set of forbidden transitions consists of all pairs of adjacent edges that have the same color. Although an s,t-PECP can be determined in polynomial time [46] in graphs, the same problem is $\mathcal{NP}$-complete in digraphs [43]. Similar complexity results follow also for the shortest s,t-PECP problem: it can be solved in polynomial time in graphs while being $\mathcal{NP}$-hard in networks.

An interesting extension occurs when changing colors is allowed with a fixed cost on two consecutive arcs with different colors when a fixed reload cost is considered in addition to the usual arc costs [47, 48]. It is associated with a transition through a vertex via two consecutive arcs of different color. Reload costs appear typically in transportation networks where each carrier is associated with a different color. Hence, changing carrier at a stop means changing arc colors at a vertex and load / unload cost of the freight becomes in fact the reload cost of the color change. They show that although the shortest s,t-PECP problem can be solved in polynomial time, one-to-many shortest PECP problem is not even approximable within $f(n) > 2$ on a network with n vertices unless $\mathcal{P} = \mathcal{NP}$, where $f(n)$ is any polynomial time computable function.

3.4. Our Contributions

As it can be noticed most of the above mentioned works report complexity results on the variants of FPCP. Except [48], all of the shortest path related studies deal with s,t-paths and solution algorithms are proposed for polynomially solvable particular cases. Only the [48] provides results on the one-to-many shortest path problem, but

it is for networks with colored arcs and reports only complexity results on the one-to-many shortest PECP. As a result, we could dare to claim that our work presents the first three special purpose exact solution algorithms for SPC. It can be used for the $\mathcal{NP}$-hard one-to-one shortest path problems with forbidden pairs and transitions, and with proper edge colors. For that purpose, a BB based on the conflicts (Chapter 6), a depth first search diving algorithm (Chapter 7), and a BB based on the stable sets (Chapter 8) are developed and their different variations are extensively tested. It is possible to say that they are quite efficient and compete in various aspects with a state-of-the-art commercial software as a consequence of the extensive computational results reported in Chapter 9.

An article consisting of the fundamentals of SPC and the BB based on conflicts named "An Efficient Branch-and-bound Algorithm for the One-to-many Shortest Path Problem with Additional Disjunctive Conflict Constraints" is currently accepted and undergoing minor revisions in *European Journal of Operations Research*. Some of the figures and material are re-used in this thesis in compliance with the following policy of the publisher: "Authors can include their articles in full or in part in a thesis or dissertation for non-commercial purposes."

4. FORMULATION

In this chapter first the formulations of SP and *Maximum Weight Stable Set Problem* (MWS) are provided since they are closely related problems with SPC. SP is important because it is the original problem to which additional conflict relations are added. MWS is important because of the stable set constraints which define the conflict relations of SPC. They are followed by *Mixed Integer Linear Programming* (MILP) and combinatorial optimization formulations of SPC.

4.1. The Shortest Path Problem

MCF formulation of the ordinary SP is given as [18]

$$\text{SP: min} \quad \sum_{a \in A(N)} c_a f_a \tag{4.1}$$

$$\text{s.t.} \quad \sum_{a \in \delta_N^+(i)} f_a - \sum_{b \in \delta_N^-(i)} f_b = \begin{cases} |V(N)| - 1 & \text{if} \quad i = s \\ -1 & \text{if} \quad i \in V(N) \setminus \{s\}. \end{cases} \tag{4.2}$$

In this formulation, continuous flow variables f_a for $a \in A(N)$, denote the amount of flow on arc a, where c_a is the unit cost of arc a. Objective function (4.1) and flow balance constraints (4.2) constitutes the linear programming formulation of one-to-many SP. Note that flow balance constraints (4.2) can be replaced by

$$\sum_{a \in \delta_N^+(i)} f_a - \sum_{b \in \delta_N^-(i)} f_b = \begin{cases} 1 & \text{if} \quad i = s \\ 0 & \text{if} \quad i \in V(N) \setminus \{s, t\} \\ -1 & \text{if} \quad i = t \end{cases} \tag{4.3}$$

if s-t shortest path problem is considered instead.

4.2. The Maximum Weight Stable Set Problem

The classical binary integer programming formulation for MWS is given in the seminal paper [49], which preceded extensive research on stable sets, as

$$\text{MWS: max} \quad \sum_{i \in V(G)} c_i x_i \tag{4.4}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 \qquad \{(i,j)\} \in E(G) \tag{4.5}$$

$$x_i \in \{0,1\} \qquad i \in V(G). \tag{4.6}$$

Here c_i is the weight of vertex i, whereas x_i is the binary selection variable for vertex $i \in V(G)$; $x_i = 1$ if vertex i is selected and $x_i = 0$ otherwise. Here, objective function (4.4) is the total weights of the selected vertices. Constraints (4.5) ensure the stable set property by forcing the selection of non-adjacent vertices, and (4.6) are for the binary restriction on x_i variables. Two other variants of stable set constraints (4.5) in the literature [50] can be written as

$$\sum_{i \in K} x_i \leq 1 \qquad K \in \mathcal{K}(G) \tag{4.7}$$

$$\sum_{i \in N_G(j)} x_i + d_G(j) x_j \leq d_G(j) \qquad j \in V(G), \tag{4.8}$$

where $\mathcal{K}(G)$ is the family of maximal cliques of graph G. Inequalities (4.7) and inequalities (4.8) are proposed respectively in [49] and [51].

It is common to consider one of (4.5), (4.7), or (4.8) to represent stable sets in formulations and they can be extended furthermore by means of various valid inequalities [50–52]. They can be interchangeably used in MILP formulation of SPC, which is explained in more detail in Section 4.3.1.

4.3. One-to-many Shortest Path Problem with Conflicts

The aim of solving SPC on network N is to find a shortest path tree (SPT) $T^* = (V(T^*), A(T^*))$ rooted at $s \in V(N)$ such that no two arcs of T^* are in conflict with each other, i.e., a conflict free shortest path tree rooted at s. Please note that the objective value of SPC is not the total cost of every arc $a \in A(T^*)$, but it is the total cost of each path from s to every other vertex $v \in V(N)$. It is possible to suggest several MILP and combinatorial optimization formulations for SPC.

4.3.1. Mixed Integer Linear Programming Formulations

SPC can be formulated as a MILP which can be obtained by adding conflict inequalities to the MCF formulation of the ordinary SP [18]. In addition to flow variables f_a, binary arc selection variables x_a for $a \in A(N)$ are required for constructing a conflict free SPT. Hence, the MILP formulation of SPC can be obtained from the formulation of SP as

$$\text{SPC: min} \quad \sum_{a \in A(N)} c_a f_a \tag{4.9}$$

$$\text{s.t.} \quad \sum_{a \in \delta_N^+(i)} f_a - \sum_{b \in \delta_N^-(i)} f_b = \begin{cases} |V(N)| - 1 & \text{if} \quad i = s \\ -1 & \text{if} \quad i \in V(N) \setminus \{s\} \end{cases} \tag{4.10}$$

$$x_a + x_b \leq 1 \qquad a \in A(N), \quad b \in N_C(a) \tag{4.11}$$

$$0 \leq f_a \leq (|V(N)| - 1)x_a \qquad a \in A(N) \tag{4.12}$$

$$x_a \in \{0, 1\} \qquad a \in A(N). \tag{4.13}$$

Objective function (4.9), which is to be minimized, is the total cost of the paths and constraints (4.10) are the flow balance equalities, which guarantee the existence of a path from the source vertex s to other vertices. We refer to packing constraints (4.11) as conflict constraints since they ensure that two arcs cannot be selected simultaneously to be on a SPT, if they are in conflict with each other. In other words, shortest path tree of SPC is conflict free. Constraints (4.12) make sure that an unselected arc cannot be on SPT. Observe that the upper bounds on the flow variables f_a are redundant if $x_a = 1$. Without conflict constraints (4.11) and with $f_a \geq 0$ in constraints (4.12) the above MILP formulation becomes the MCF formulation of the ordinary SP. Observe that inequalities (4.11) and (4.13) describe together the stable sets of conflict graph C, and equalities (4.10) with the non-negativity restrictions $f_a \geq 0$, $a \in A(N)$ represent the spanning outtrees of N. As a result, SPC aims to determine a *conflict free SPT*, which is a spanning outtree of N whose arcs form a stable set of C of size $|V(N)| - 1$.

It is possible to obtain two more SPC formulations by replacing inequalities (4.11) with (4.7) and (4.8) after adopting for conflict graph C. One of them follows from the

replacement of inequalities (4.11) with

$$\sum_{b \in N_C(a)} x_b + d_C(a) x_a \leq d_C(a) \qquad a \in A(N), \tag{4.14}$$

which are obtained by aggregating inequalities (4.11) for each $a \in A(N)$ on its neighbours b in C, i.e., $N_C(a)$, as suggested in [51]. Clearly, $d_C(a) = |N_C(a)|$, which is the degree of vertex a in C. We refer to this version as the *weak formulation* (W), and the previous one as the *strong formulation* (S). Conflict inequalities (4.11) can also be replaced with

$$\sum_{a \in K} x_a \leq 1 \qquad K \in \mathcal{K}(C), \tag{4.15}$$

which yields so-called *clique formulation* (K). Here, $\mathcal{K}(C)$ is the family of maximal cliques of conflict graph C and inequalities (4.15) allow the inclusion of at most one vertex from each one of them. If we let P_S, P_W and P_K denote the polyhedra representing the feasible solution sets of the LP relaxations of the strong, weak and clique formulations, then $P_K \subseteq P_S \subseteq P_W$ holds since inequalities (4.14) are obtained by aggregating inequalities (4.11) for $b \in N_C(a)$, and inequalities (4.15) define the facets of the convex hull of the stable sets, i.e., the facets of the stable set polytope [49].

4.3.2. Combinatorial Optimization Formulation

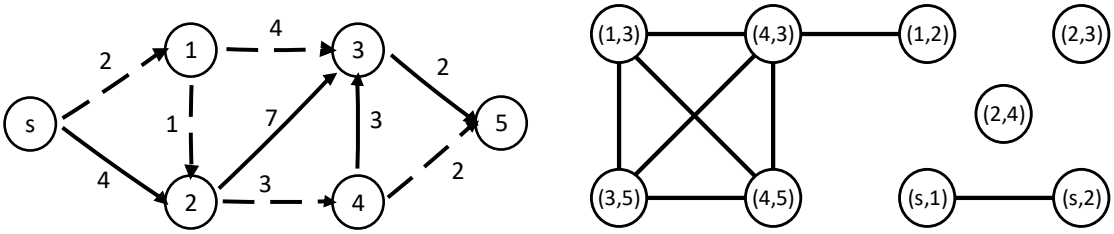
Let $\mathcal{S} \subseteq A(N)$ be a stable set of conflict graph C. Let $N[\mathcal{S}]$ be a sub-network induced by arc subset $\mathcal{S}$ (vertices of C correspond to arcs of N), which includes at least one directed spanning outtree rooted at s. Let Σ be the family of such $\mathcal{S}$. Let $z(\mathcal{S})$ be the cost of the shortest spanning outtree in $N[\mathcal{S}]$. Then, the combinatorial optimization formulation of SPC can be stated as

$$\min \{z(\mathcal{S}) : \mathcal{S} \in \Sigma\}. \tag{4.16}$$

It easily follows that since SP can be solved in polynomial time, SPC can be solved in polynomial time if there are polynomially many such subsets and they can be generated in polynomial time. This is only possible if C follows a specific structure as a special case for SPC.

4.3.3. An Illustrative Example

An illustration is provided in Figure 4.1. Figure 4.1a and b respectively represent a sample network with arc weights and conflict graph. When SPC is relaxed with respect to conflict constraints we obtain the relaxed optimal solution which is the shortest path tree consisting of arcs $\{(s,1), (1,2), (2,4), (1,3), (4,5)\}$ with total cost 25 (sum of all shortest path distances). Dashed arcs show this relaxed solution in Figure 4.1a. As can be seen from the conflict graph in Figure 4.1b, $\{(1,3), (3,5), (4,3), (4,5)\}$, $\{(s,1), (s,2)\}$ and $\{(1,2),(4,3)\}$ are in conflict. The first subset forms the maximum clique of C and a conflict free spanning outtree of N can include at most one of its vertices. $\{(s,1), (1,2), (2,3), (2,4), (4,5)\}$ is the arc set of an optimal conflict free spanning outtree with total cost of 29, higher than the relaxed solution cost. Note that these conflict free spanning outtree arcs constitute a stable set of C on conflict graph in Figure 4.1b of cardinality 5, which is one less than the number of vertices of the network.



(a) (b)

Figure 4.1. (a) Sample network and (b) its conflict graph.

5. UPDATING A SHORTEST PATH TREE AFTER ARC DELETION

As explained in the previous chapter, SP has a MCF formulation (4.1) – (4.2) without x_a variables and related constraints, i.e., conflict inequalities (4.11), upper bounds in (4.12) and binary restrictions (4.13) of the MCF formulation of SPC. Besides, all vertices have unit demand except s, which is the only source vertex with supply $|V(N)| - 1$. Due to these special features an optimal solution is a SPT with vertices $V(T)$ and arcs $A(T)$, and has the following properties [18]:

(i) It is a spanning outtree. Hence, $V(T) \equiv V(N)$, $|A(T)| = |V(N)| - 1$, and there is exactly one inarc, i.e., $d_T^-(v) = 1$, for $v \in V(T)$, except the source vertex s, which is the root of the tree with indegree zero, i.e., $d_T^-(s) = 0$;

(ii) It is non-degenerate, i.e., flow variables associated with the arcs of the tree are positive, i.e., $f_a > 0$ for $a \in A(T)$;

(iii) Every tree arc has zero reduced cost, i.e., $\bar{c}_{ij} = c_{ij} - \pi_i + \pi_j = 0$ for $(i, j) \in A(T)$. Here, π_i and π_j are the dual variables or potentials associated with vertices i and j. The reduced cost of the nontree arcs are non-negative;

(iv) Vertex potentials π_i, $i \in V(T)$ are equal to the negative of the shortest distances from source s to the vertex i;

(v) Optimal flow value on an arc is equal to the number of vertices in the subtree rooted at its head.

Many network optimization problems in the literature have algorithms that requires successive solution of SP [18]. If SP is solved after relaxing the conflict constraints on SPC, then a SPT, which can be infeasible with respect to conflict constraints of SPC but optimal in terms of the shortest path distances to all vertices from the source, is generated. Let us consider one such violation between two arcs a and b on a SPT. If the problem is divided into two with new arc sets i.e., $A_1(N) = A(N) \setminus a$ and $A_2(N) = A(N) \setminus b$, one could get rid of one cause of infeasibility generated by

solving the SP in the first place. Since SP is polynomially solvable, successive solution of SP in order to obtain a solution of SPC eliminating violations as they appear constitutes a crude algorithm for SPC. However, as lots of SP sub-problems would be potentially generated, solving one such sub-problem as efficient as possible is very important for the efficiency of the overall algorithm. Potential reoptimization techniques are discussed for successively generated sub-problems in this chapter.

Various methods for the reoptimization of the shortest path tree have been proposed. They are more general in the sense that they consider cost or root changes [53–58]. Since the deletion of an arc is the only atomic operation that must be repeated during branching in the BB algorithms introduced in the following chapters, we focus on updating a shortest path tree after deleting one of its arcs.

5.1. A Primal-dual Method

When an arc is deleted from SPT, it is divided into two subtrees. Let $T = (V(T), A(T))$ be a SPT obtained on a network $N = (V(N), A(N))$ rooted at vertex s, and $T_1 = (V(T_1), A(T_1))$ and $T_2 = (V(T_2), A(T_2))$ be the two subtrees created by the removal of tree arc $a = (i, j) \in A(T)$ such that $s \in V(T_1)$. Also, let $S \equiv V(T_1)$ and $\overline{S} \equiv V(T_2)$ be the vertices of the subtrees, and $(S, \overline{S})$ and $(\overline{S}, S)$ denote the sets of arcs from S to $\overline{S}$, i.e., forward arcs, and from $\overline{S}$ to S, i.e., backward arcs, in the cut $[S, \overline{S}]$. Because T is a spanning outtree, $i \in S$ and $j \in \overline{S}$, i.e., $(i, j) \in (S, \overline{S})$. Besides, since T is a shortest path tree the reduced costs of the tree arcs are zero and the reduced costs of the nontree arcs are non-negative, i.e., $\bar{c}_a = 0$ for $a \in A(T)$ and $\bar{c}_a \geq 0$ for $a \in A(N) \setminus A(T)$. Hence, $\bar{c}_{ij} = 0$ and $\bar{c}_{kl} \geq 0$ for $(k, l) \in (S, \overline{S}) \setminus \{(i, j)\}$. Besides, subtree T_1 represents the shortest paths from s to any other vertex $u \in V(T_1)$. Thus, it is a *partial* shortest path tree rooted at s. The partition created by the removal of arc $a = (i, j)$ is illustrated in Figure 5.1.

Let us construct a new spanning outtree rooted at s by adding new arcs to T_1 satisfying both primal and dual feasibility, and complementary slackness conditions.

We first determine nontree arc $a' = (k, l) \in (S, \overline{S})$ such that $\bar{c}_{kl} = \min_{(u,v)\in(S,\overline{S})}\{\bar{c}_{uv}\}$. Unless $l = j$, where j is the original head vertex of removed arc $a = (i, j)$, the addition of $a' = (k, l)$ to $A(T_1) \cup A(T_2)$ creates a spanning tree that is not an outtree. There is an arc $(p, l) \in A(T_2)$ and primal feasibility is violated due to two incoming arcs to l with the addition of $a' = (k, l)$. Hence, (p, l) should be removed as an attempt to restore primal feasibility. Note that the addition of arc (k, l) can increase the size of $V(T_1)$ more than one vertex since it links a subtree, rooted at vertex l. Let us denote the vertices of the newly added subtree as S^{new}. As the subtree is attached to T_1 and detached from T_2, S and $\overline{S}$ becomes respectively $S \cup S^{new}$ and $\overline{S} \setminus S^{new}$. If $a_2 = (p, l) \in A(T_2)$ is selected as the new removed arc, the whole process could be repeated until $A(T_1)$ spans network N, i.e., $|A(T_1)| = |V(N)| - 1$, and $A(T_2) = \emptyset$.

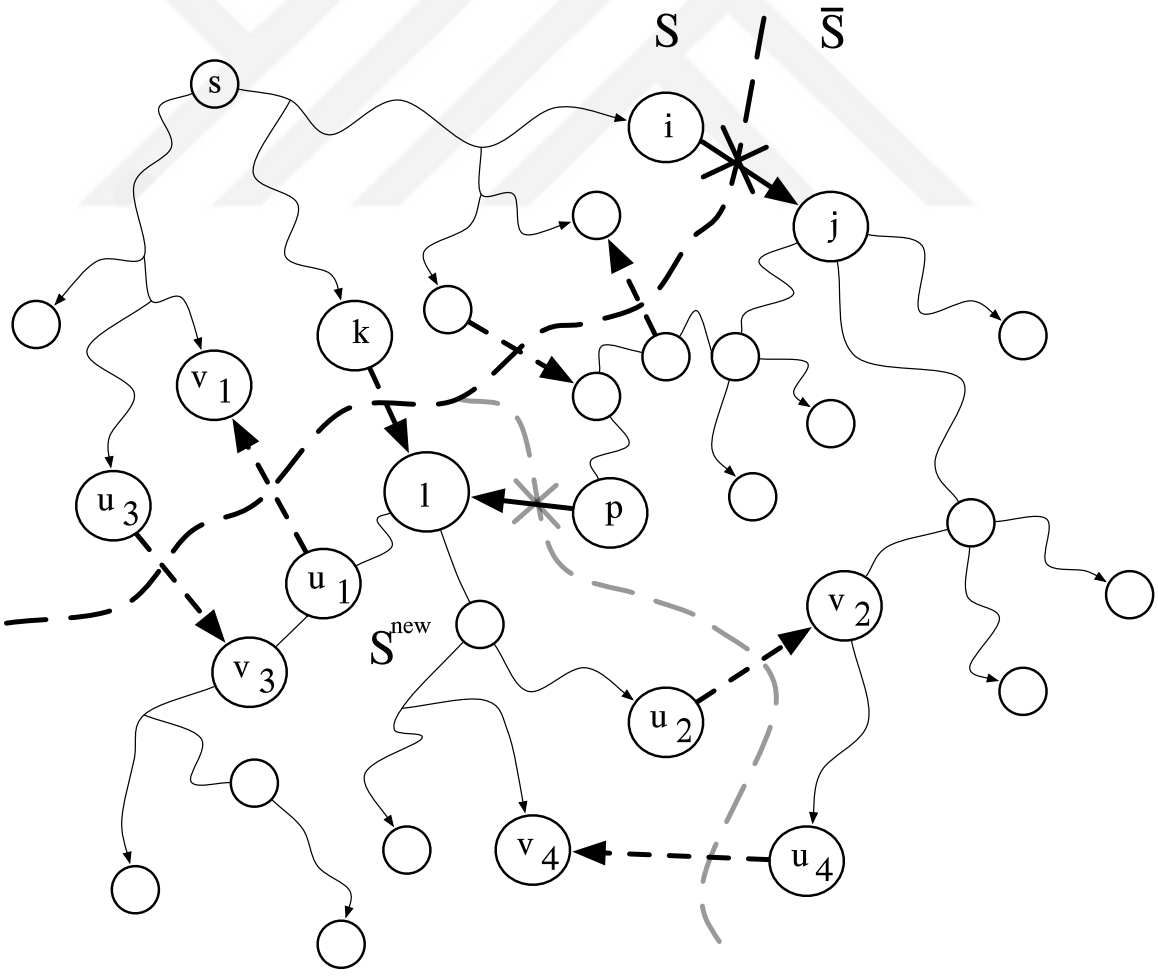


Figure 5.1. Illustration of tree partition with the removal of arc (i, j).

Since $\bar{c}_{kl} \geq 0$, complementary slackness can be violated for (k, l) after making it a tree arc. However, it is possible to make it zero as required by subtracting $\bar{c}_{kl}$ from potential (i.e., dual value) of π_l. The new reduced cost clearly becomes $c_{kl} - \pi_k + (\pi_l - \bar{c}_{kl}) = \bar{c}_{kl} - \bar{c}_{kl} = 0$. In order to prevent the chain effect of the potential change, the potentials of the vertex set S^{new} of the newly added subtree, which is the subtree of T_2 rooted at vertex l, are decreased by the same amount $\bar{c}_{kl}$. Although, this decrease does not affect the reduced costs of the arcs with both ends in S^{new} and S, the reduced costs of the arcs with one end in S^{new} change as follows:

(i) The reduced costs of the arcs from S^{new} to $S \bigcup \overline{S} \setminus S^{new}$ increase by $\bar{c}_{kl}$. Since this value is non-negative no negative reduced cost can result because of this update. Referring to Figure 5.1, (u_1, v_1) and (u_2, v_2) are two of such arcs.

(ii) Reduced costs of the arcs from S to S^{new} decrease by $\bar{c}_{kl}$. Since these arcs belong to $(S, \overline{S})$ and $\bar{c}_{kl} = \min_{(u,v)\in(S,\overline{S})}\{\bar{c}_{uv}\}$, no negative reduced cost can result because of this update as well. As can be seen from Figure 5.1 (u_3, v_3) is one of such arcs.

(iii) Reduced costs of arcs from $\overline{S} \setminus S^{new}$ to S^{new}, e.g., (u_4, v_4) of Figure 5.1, decrease by $\bar{c}_{kl}$. If $\overline{S} \setminus S^{new} \neq \emptyset$ (or equivalently $l \neq j$), then there can be arcs with heads in S^{new} and tails in $\overline{S} \setminus S^{new}$ whose reduced costs become negative after the updates. In fact, the next arc to be removed i.e., (p, l), is one of them if $\bar{c}_{kl} > 0$. Before the updates $\bar{c}_{pl} = 0$, since it is in the shortest path tree T and satisfy complementary slackness conditions. Therefore, decreasing the potential π_l by $\bar{c}_{kl}$ makes the new reduced cost $-\bar{c}_{kl}$.

As it can be noticed dual feasibility can be violated only in the third case for the arcs with heads in S^{new} and tails in $\overline{S} \setminus S^{new}$. They are back arcs from $\overline{S} \setminus S^{new}$ to S^{new}, and they are not within the set of potential arcs to be included in the new shortest path tree. They can be prevented from further considerations. Notice that T_1 is enlarged attaching the subtree rooted at l by means of (k, l), and detached from T_2 by removing (p, l) in order to restore primal feasibility as illustrated also in Figure 5.1. The whole process is repeated after setting the new leaving arc (i, j) as (p, l), until the growing T_1 spans all vertices of network N.

Backward arcs from $\overline{S}$ to S are not taken into consideration because their addition results in another inarc for a vertex in S. In short, they are not required in the construction of the new shortest path tree. The pseudo-code of the primal-dual method for updating a SPT after the deletion of a tree arc (i, j) is given in Figure 5.2. In case more than one tree arcs have to be deleted, it can be called for each arc within a loop.

Input: T : A shortest path tree with root s, (i, j): The arc to be removed

Output: New shortest path tree rooted at s, or infeasibility detection

begin

1: Determine the subtrees T_1, T_2, and prepare sets $S, \overline{S}$

2: Remove all backward arcs in $(\overline{S}, S)$

3: **while** $|T_1| < |T|$ **and** $(S, \overline{S}) \not\equiv \emptyset$ **do**

4: $(k, l) \leftarrow \arg\min_{(u,v)\in(S,\overline{S})}\{\bar{c}_{uv}\}$ ▷ *minimum reduced cost arc from S to $\overline{S}$*

5: $\bar{c}_{kl} \leftarrow \min_{(u,v)\in(S,\overline{S})}\{\bar{c}_{uv}\}$ ▷ *minimum reduced cost of the arc from S to $\overline{S}$*

6: Determine the subtree and its vertex set S^{new}

7: $\pi_u \leftarrow \pi_u - \bar{c}_{kl}$ for $u \in S^{new}$

8: Update the reduced costs as:

9: • Arcs from S to S^{new}: decrease the reduced costs by $\bar{c}_{kl}$

10: • Arcs from S^{new} to $S \bigcup \overline{S} \setminus S^{new}$: increase the reduced costs by $\bar{c}_{kl}$

11: • Arcs from $\overline{S} \setminus S^{new}$ to S^{new}: decrease the reduced cost by $\bar{c}_{kl}$

12: Transfer the subtree of T_2 rooted at vertex l, to T_1 by inserting arc (k, l)

13: $S \leftarrow S \bigcup S^{new}$

14: $\overline{S} \leftarrow \overline{S} \setminus S^{new}$

15: $(i, j) \leftarrow (p, l) \in T_2$ and remove it from T_2

16: Remove all backward arcs from $\overline{S}$ to S^{new} ▷ *(i, j) is also removed*

17: **if** $|T_1| = |T|$ **then**

18: **return** T_1 ▷ *T_1 is the new shortest path tree*

19: **else**

20: **return Infeasible Sub-problem** ▷ $(S, \overline{S}) \equiv \emptyset$

end

Figure 5.2. Primal-dual algorithm.

Does the proposed method for reoptimization always return a feasible solution? First of all, since at least an arc is removed from the tree, the new SP could be infeasible. For example, a vertex without remaining inarcs after the deletion is a potential cause of infeasibility. This can be detected prior to reoptimization using an infeasibility test detailed in Section 6.6. The correctness and the computational complexity of primal-dual is discussed in Section 5.2 and Section 5.3 respectively.

5.2. Correctness

Primal-dual algorithm (Figure 5.2) grows a partial SPT rooted at s. Given the arc to be deleted from the shortest path tree, which we denote as (i, j) initially, new tree arc (k, l) is determined, arc (p, l) is removed from T, and the subtree rooted at vertex l is attached to the partial shortest path tree T_1. We next show that the algorithm finds new SPT in finite steps if SP is feasible.

Proposition 5.1. *Algorithm terminates in finite steps where entering arc is always from S to S^{new} increasing the size of S until restoration or infeasibility detection.*

Proof. If entering arc is $\{(k, j) : k \in S, j \in \overline{S}\}$, then j is the root of T_2 and $S^{new} = \overline{S}$ and $\overline{S} \setminus S^{new} = \emptyset$, then tree is restored as (k, j) enters. For a network with finite number of vertices, either T_2 becomes smaller with each iteration until $l = j$ and the algorithm terminates, or $(S, \overline{S})$ is empty in interim steps and infeasibility is detected. In both cases algorithm terminates in finite steps. □

Proposition 5.2. *An entering arc (k, l) can never leave T_1.*

Proof. Previously entered arcs are in T_1 having both their tails and heads in S. Each leaving arc (p, l) is from $\overline{S}$ to S^{new}, arcs in T_1 can not leave. □

Proposition 5.3. *Entering arc (k, l) with reduced cost $\bar{c}_{kl}$ reduces the vertex potentials of vertices in S^{new} by $\bar{c}_{kl}$. The new reduced cost of arc (k, l) is zero.*

Proof. Reduced cost is defined as $\bar{c}_{kl} = c_{kl} - \pi_k + \pi_l^{old}$ where π_l^{old} is the old potential of vertex $l \in S^{new}$. Due to step 7 in the algorithm (Figure 5.2) the new potential is $\pi_l = \pi_k - c_{kl} = \pi_l^{old} - \bar{c}_{kl}$. The new reduced cost of this arc will be $\bar{c}_{kl}^{new} = c_{kl} - \pi_k + \pi_l^{old} - \bar{c}_{kl} = 0$. □

Proposition 5.4. *When the algorithm terminates with a solution, i.e.,* $|T_1| = |V(N)| - 1$*, all reduced costs on* T_1*, the re-optimized tree, are zero.*

Proof. Due to Proposition 5.2, S has permanent labels, and due to Proposition 5.3 all reduced costs are zero on T_1. □

Proposition 5.5. *The reduced cost* $\bar{c}_{k,l}$ *of minimum reduced cost arc* (k, l) *in* $(S, \overline{S})$ *is non-decreasing during iterations.*

Proof. If $\overline{S} \setminus S^{new}$ is not empty the algorithm continues with another iteration. Additional arcs in $(S, \overline{S})$ for the next iteration are from S^{new} to $\overline{S} \setminus S^{new}$. With the update of vertex potential of S^{new}, these non-negative reduced costs are increased by the minimum reduced cost at each iteration. Therefore, the next minimum reduced cost in $(S, \overline{S})$ will be no less than the previous minimum reduced cost. □

Proposition 5.6. *When the algorithm terminates with a solution, all reduced costs are non-negative.*

Proof. Updating the labels of vertices in S^{new} can reduce the reduced costs of inarcs of S^{new} by the minimum reduced cost, i.e., $\bar{c}$. If an arc is from S to S^{new}, this reduction can not create a negative cost arc, since the minimum reduced cost arc is selected from $(S, \overline{S})$. From $\overline{S} \setminus S^{new}$ to S^{new} all reduced costs decrease by $\bar{c}$. This can make a reduced cost negative. However, this reduction can only happen if $\overline{S} \setminus S^{new} \neq \emptyset$ (not the last iteration) and it is the only reduction made on its reduced cost throughout the iterations of the algorithm. Let arc (u, v) be such an arc on iteration t_1 where its reduced cost is reduced by $\bar{c}$. After sets S and $\overline{S}$ are updated at the end of iteration t_1, $u \in \overline{S}$ and $v \in S$. Due to Proposition 5.1, tail vertex u of arc (u, v) is an element of S^{new},

i.e., $u \in S^{new}$, in a future iteration $t_2 > t_1$ if algorithm terminates with a new SPT. In iteration t_2, (u, v) is from S^{new} to S and increase its reduced cost by the minimum reduced cost on $(S, \overline{S})$ at iteration t_2. Due to Proposition 5.5, this increase is higher than previous reduction making it non-negative when the algorithm terminates. □

Proposition 5.7. *There is a SPT when the algorithm terminates with a solution.*

Proof. Follows from Proposition 5.4 and Proposition 5.6. □

Therefore, the primal-dual algorithm given in Figure 5.2 either stops with a spanning outtree rooted at s satisfying primal and dual feasibility, and complementary slackness conditions, which makes it a SPT, or detects infeasibility in finite steps. An illustrative application of this algorithm inside a BB can be found in Section 6.5.3.

5.3. Computational Complexity

In order to study the computational complexity of the algorithm, let us introduce the iteration subscript r in order to identify the evolution of the generated subsets of vertices $S_r, \overline{S}_r, S_r^{new}$ representing the respective subsets at a specific algorithm iteration r. Let (k_r, l_r) be the entering arc at iteration r. Then we can use a candidate arc list Q_r in the determination of new tree arc (k_r, l_r). It consists of a subset of forward arcs in $(S_r, \overline{S}_r)$. Each arc in Q_r has the smallest reduced cost within the arcs of $(S_r, \overline{S}_r)$ having the same head vertex. In other words, if $(u, v) \in Q_r$, then $\bar{c}_{uv} = \min_{(w,v)\in(S_r,\overline{S}_r)}\{\bar{c}_{wv}\}$. Q_1 is constructed while preparing $(S_1, \overline{S}_1)$ and Q_{r-1} is updated while preparing $(S_r, \overline{S}_r)$ to obtain Q_r.

Proposition 5.8. *The worst case complexity of the primal-dual algorithm (Figure 5.2) is* $\mathcal{O}(|V(N)|^2 + |A(N)|)$.

Proof. Major cost items of the primal-dual algorithm and their worst case complexity can be listed as follows:

(i) Determination of subtrees and potential updates: Initially S_1 is constructed from subtree T_1. Then, S_r^{new} is obtained at step r. All these sets are disjoint and $\{S_r^{new}\}_{r=1}^{h}$ clearly forms a partition of $\overline{S}_1$, where h is the last iteration. This can be done by traversing each subtree using graph search, which is linear in the size of the vertex subsets. In the meantime it is possible to update vertex potentials. In other words, total cost associated with these operations is $|S_1| + \sum_{r=1}^{h} |S_r^{new}|$, which is $\mathcal{O}(|V(N)|)$. Here, h denotes the number of iterations.

(ii) Determination of arcs $(S_r, \overline{S}_r)$: It is enough to handle the outarcs of the vertices in S_1 in order to construct $(S_1, \overline{S}_1)$. For the construction of $(S_r, \overline{S}_r)$ for $r = 2, 3, \ldots, h$, first the inarcs of the vertices in S_r^{new} are handled in order to delete the arcs in (S_{r-1}, S_r^{new}) from $(S_{r-1}, \overline{S}_{r-1})$. Notice that $S_r^{new} \subseteq \overline{S}_{r-1}$. Then, the outarcs of the vertices in S_r^{new} are handled to identify the arcs of $(S_r^{new}, (\overline{S}_{r-1} \setminus S_r^{new}))$, which are added to $(S_{r-1}, \overline{S}_{r-1})$ in order to form $(S_r, \overline{S}_r)$. Total cost associated with these operations is

$$\sum_{v \in S_1} d_N^+(v) + \left[\sum_{r=1}^{h} \sum_{v \in S_r^{new}} d_N^-(v) + \sum_{r=1}^{h} \sum_{v \in S_r^{new}} d_N^+(v) \right] = 2|A(N)| - \sum_{v \in S_1} d_N^-(v),$$

which is $\mathcal{O}(|A(N)|)$ in the worst case.

(iii) Update of Q_r: The outarcs of the vertices in S_1 are handled for preparing the initial Q_1, the inarcs of the vertices in S_r^{new} are handled in order to delete related entries from Q_r, and the outarcs of the vertices in S_r^{new} are handled in order to determine the new entries for Q_r. At sum, this requires the same amount of work as the determination of $(S_r, \overline{S}_r)$ does, which is $\mathcal{O}(|A(N)|)$ in the worst case.

(iv) Determination of arc (k_r, l_r): First of all by definition of Q_r, $|Q_r| = |\overline{S}_r|$. Besides, $|\overline{S}_r|$ decreases as much as $|S_r^{new}|$ at step r. Since $|S_r^{new}| \geq 1$, $|\overline{S}_1| \leq |V(N)| - 1$ and the smallest entry of Q_r can be determined in $|Q_r|$ operations, total cost of these operations is $\sum_{r=1}^{h} |\overline{S}_r| \leq \frac{|V(N)|(|V(N)|-1)}{2}$, and the worst case cost associated with the determination of (k_r, l_r) for $r = 1, 2, \ldots, h$ becomes $\mathcal{O}(|V(N)|^2)$.

Therefore, by summing all these ingredients together makes $\mathcal{O}(|V(N|^2)| + |A(N)|)$. $\square$

A couple of comments can be made about the computational complexity of the primal-dual algorithm. Proposition 5.8 says that it is equal to the worst case complexity of Dijkstra's algorithm [59]; but this is quite extreme since it holds under the conditions $|\overline{S}_1| = |V(N)| - 1$ and $|S_r^{new}| = 1$ for $r = 1, \ldots, h$. Hence, the practical performance of the algorithm must be remarkably better. As it is the case with Dijkstra's algorithm, data structure used for Q_r has a direct effect on the efficiency, and the worst case complexity becomes $\mathcal{O}(|V(N|)| \log |V(N)| + |A(N)|)$ when Q_r is represented by a Fibonacci heap [60].

Shortest path reoptimization versus resolution from scratch has been around as an interesting research question for a while. Although, there is no clear cut answer, the recent experimental study provides some insights [58]. Their aim is to identify graph characteristics which makes reoptimization more beneficial than resolution from scratch performed with Dijkstra's algorithm. According to their results, reoptimization is more suitable when the perturbed network is general and does not have an intrinsic structure, and the number of deleted SPT arcs are considerably smaller than the number of network arcs, which is also observed in an earlier computational study [56]. One of the main conclusions is that although dynamic algorithms and Dijkstra's algorithm have similar worst case time complexity, it is beneficial to choose dynamic ones especially when the ratio of modified SPT arcs, which we translate as deleted shortest path tree arcs in our case, is relatively small.

6. A BRANCH AND BOUND ALGORITHM BASED ON CONFLICTS

At node t of the BB tree, we let SPC$^{(t)}$ denote the corresponding sub-problem. At an abstract level, SPC$^{(t)}$ can be represented by the triplet $(I^{(t)}, X^{(t)}, F^{(t)})$ partitioning the set of arcs $A(N)$. $I^{(t)}$ and $X^{(t)}$ are respectively the subset of arcs which must be *included in* and *excluded from* the shortest path tree $T^{(t)}$, which is an optimal solution of the relaxed sub-problem $SP^{(t)}$. On the other hand, $F^{(t)}$ is the subset of *free* arcs which have not been added to any one of $I^{(t)}$ or $X^{(t)}$ yet. In other words, $f_a > 0$ and $x_a = 1$ for $a \in I^{(t)}$, $f_a = 0$ and $x_a = 0$ for $a \in X^{(t)}$ and $f_a \geq 0$ and $x_a \in \{0, 1\}$ for $a \in F^{(t)} \equiv A(N) \setminus I^{(t)} \cup X^{(t)}$. Hence, $F^{(t)}$ represents the subset of arcs that are candidate to be added to either $I^{(t)}$ or $X^{(t)}$ of the newly created sub-problems during branching. Besides, none of the arcs in $I^{(t)}$ is conflicting with each other and none of the arcs in $I^{(t)}$ is conflicting with any arcs in $F^{(t)}$, since when a new arc is added to $I^{(t)}$ all arcs in $F^{(t)}$ conflicting with it are excluded. However, relative to the conflict relations represented with conflict sub-graph $C^{(t)} = (V(C^{(t)}), E(C^{(t)}))$, some of the arcs in $F^{(t)}$ can be in conflict. At the same time, since $A(N) \equiv V(C)$, the triplet $(I^{(t)}, X^{(t)}, F^{(t)})$ of the arc set $A(N)$ partitions also $V(C)$. As a result, $F^{(t)} \equiv V(C^{(t)}) \equiv V(C) \setminus (I^{(t)} \cup X^{(t)}) \equiv A(N) \setminus (I^{(t)} \cup X^{(t)})$, is satisfied.

6.1. Relaxations of the Sub-problems

Sub-problem SPC$^{(t)}$ consists of the determination of a conflict free shortest path tree that includes the arc subset $I^{(t)}$ on the spanning subnetwork $N^{(t)} = (V(N^{(t)}), A(N^{(t)}))$ with $V(N^{(t)}) = V(N)$ and $A(N^{(t)}) = I^{(t)} \cup F^{(t)}$. Meanwhile, the relaxed sub-problem SP$^{(t)}$ consists of the determination of a SPT, $T^{(t)} = (V(N^{(t)}), A(T^{(t)}))$, with $I^{(t)} \subseteq A(T^{(t)})$ and $(A(T^{(t)}) \setminus I^{(t)}) \subseteq F^{(t)}$. Notice that this is not exactly the ordinary one-to-many shortest path problem on $N^{(t)}$, since an optimal solution is restricted to include $I^{(t)}$. Thus, the solution of SP$^{(t)}$ requires additional care.

$I^{(t)}$ of $\mathrm{SPC}^{(t)}$ can be represented as a forest of outtrees, i.e., arborescences, with vertices $V(I^{(t)})$. Whenever an arc from the set of free arcs $(u,v) \in F^{(t)}$ is added to forest $I^{(t)}$, there are four possibilities: (i) if $u \notin V(I^{(t)})$ and $v \notin V(I^{(t)})$, $(u,v) \in I^{(t)}$ is a new outtree of the forest, (ii) if $u \in V(I^{(t)})$ and $v \notin V(I^{(t)})$ or $u \notin V(I^{(t)})$ and $v \in V(I^{(t)})$, the size of an outtree in $I^{(t)}$ grows by one, (iii) if u and v are in different outtrees they are merged into a single outtree, (iv) if u and v are in the same outtree, adding (u,v) creates a cycle and hence (u,v) is not a valid arc to include. Please note that there is exactly one inarc for each vertex in SPT. Therefore, whenever arc (u,v) is included, i.e., $(u,v) \in I^{(t)}$, all other inarcs of v, i.e., $(p,v) \in F^{(t)}$, can be removed from $F^{(t)}$. Hence, the possible cycle in case (iv) is actually a circuit where $v \neq s$ is the root of a outtree in the forest.

In an optimal solution of $\mathrm{SP}^{(t)}$, conflicts can exist only between the arcs that are selected from $F^{(t)}$. $I^{(t)} \cup \{a\}$ can not contain a circuit if $a \in F^{(t)} \cap A(T^{(t)})$. That is because $A(T^{(t)}) \supseteq I^{(t)} \cup \{a\}$ and by definition there is no circuit in a tree. In other words, the inclusion of a free SPT arc cannot create a circuit. However, for $a \in F^{(t)} \setminus A(T^{(t)})$, $I^{(t)} \cup \{a\}$ can have a circuit (case (iv)) although $I^{(t)} \cup \{a\}$ is conflict free, which should be taken into account whenever an arc outside of the SPT is added to $I^{(t)}$.

6.2. Branching Rules

At parent node t of the BB search tree, we adopt three dichotomized branching rules, which are called *conflicting pair branching*, *clique branching* and *conflicting arc branching*, in order to create two child sub-problems $\mathrm{SPC}^{(ti)}$ for $i = 1, 2$ of $\mathrm{SPC}^{(t)}$ by dividing its feasible solution set into two subsets. Note that this division does not have to yield a partition. It suffices that the union of the newly created subsets yields the feasible solution set before the division.

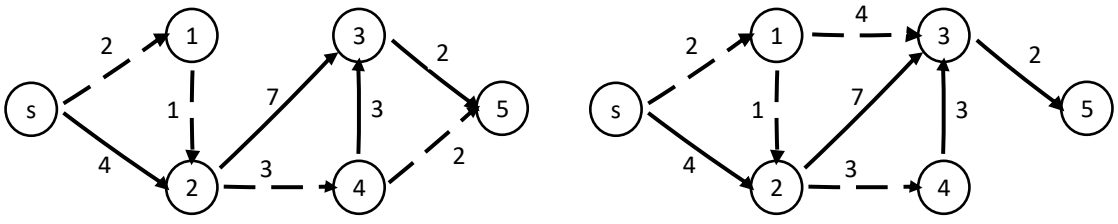
Each one of these three rules essentially aims to get rid of a conflicting arc pair that exists in a relaxed optimal solution, i.e., a SPT, of the parent node's relaxed

problem, from appearing one more time in the optimal solution of the newly created sub-problems after the division. Hence, a pair of conflicting arcs, say a_1 and a_2 in an optimal solution of the SP relaxation of sub-problem $\mathrm{SPC}^{(t)}$, i.e., $\mathrm{SP}^{(t)}$, is selected first. Then, sub-problems $\mathrm{SPC}^{(ti)}$ for $i = 1, 2$ or equivalently the triplets $(I^{(ti)}, X^{(ti)}, F^{(ti)})$ for $i = 1, 2$ are generated from $(I^{(t)}, X^{(t)}, F^{(t)})$ using a_1 and a_2 according to the considered rule.

6.2.1. Conflicting Pair Branching

This rule updates both $X^{(t)}$ and $F^{(t)}$ while not changing $I^{(t)}$. The division of the feasible solution set of a sub-problem is based on the deletion of the arcs of a selected conflicting pair. Sub-problems $\mathrm{SPC}^{(ti)}$ are generated by removing a_i from $\mathrm{SP}^{(t)}$ by setting $X^{(ti)} = X^{(t)} \cup \{a_i\}$, $F^{(ti)} = F^{(t)} \setminus \{a_i\}$ for $i = 1, 2$, respectively. Notice that we do not obtain a partition of the solution set of the parent problem when this rule is applied. For example the spanning outtree $\{(s, 1), (1, 2), (2, 3), (2, 4), (3, 5)\}$ is a feasible SP solution in both branches. That is to say, although the same conflicting arc pair does not reappear deeper in the subtree, some of the feasible solutions may appear in both branches in future iterations of BB.

We illustrate this rule in Figure 6.1 for the sample network and its conflict graph given in Figure 4.1 with the relaxed optimal solution, i.e., SPT, $\{(s, 1), (1, 2), (1, 3), (2, 4), (4, 5)\}$. Figure 6.1a and b show the two branches resulting from branching on conflicting arcs $(1, 3)$ and $(4, 5)$ of the search tree.



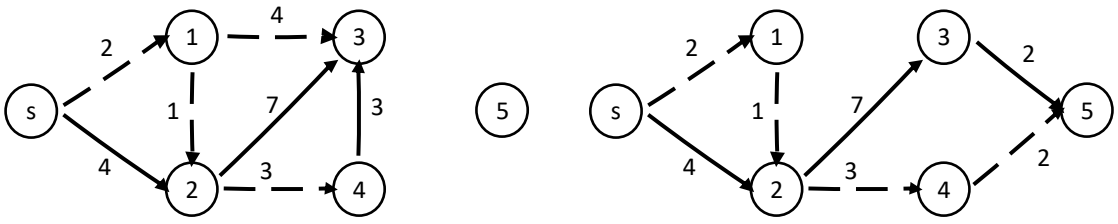
(a) (b)

Figure 6.1. Conflicting pair branching: (a) First branch, (b) second branch.

6.2.2. Clique Branching

The arcs of a conflict free SPT represents a stable set of the conflict graph, and thus at most one vertex (i.e., arc of $A(N)$) from each clique of the conflict graph can be in a feasible solution of SPC. First, we find a maximal clique (preferably with the maximum cardinality) of the conflict graph that includes the conflicting arc pair, a_1 and a_2. Then, we partition its vertices into two subsets such that a_1 and a_2 are separated. Finally, we delete the vertices of the two subsets from $N^{(t)}$ and $C^{(t)}$ to generate two child sub-problems.

Formally speaking, we find a clique K of conflict sub-graph $C^{(t)}$ with $\{a_1, a_2\} \in K$. By partitioning the clique K, we obtain two subsets K_i for $i = 1, 2$ containing arc a_i for $i = 1, 2$, respectively. Note that subsets K_1 and K_2 have equal (almost equal) cardinality when $|K|$ is even (odd). Then, we generate the two sub-problems by setting, $X^{(ti)} = X^{(t)} \cup K_i$, $F^{(ti)} = F^{(t)} \setminus K_i$ for $i = 1, 2$. Note that the determination of K with the highest possible cardinality has a potential to increase the efficiency of this branching rule. However, it is not required to find the maximum cardinality clique in $C^{(t)}$ including the conflicting arc pair.



(a) (b)

Figure 6.2. Clique branching: (a) First branch, (b) second branch.

Figure 6.2 illustrates clique branching on the sample problem of Figure 4.1. The clique $\{(1,3),(4,3),(3,5),(4,5)\}$ is divided into two equally sized subsets $\{(1,3),(4,3)\}$ and $\{(3,5),(4,5)\}$ for conflicting pair $(1,3)$ and $(4,5)$. Resulting branches can be seen in Figure 6.2a and b respectively. This rule can result in a more balanced division of the solution space. However, it requires the determination of a maximal clique that includes a conflicting pair at each iteration.

6.2.3. Conflicting Arc Branching

This rule is based on one of the conflicting arcs of the selected pair. Hence, one of a_1 or a_2 is selected first. If a_1 is selected as the branching arc, sub-problems $\{\text{SPC}^{(ti)}\}$ for $i = 1, 2$ are created by deleting the arcs conflicting with a_1 in one branch (equivalently meaning the inclusion of a_1), and deleting a_1 at the other branch from $\text{SPC}^{(t)}$, respectively. This is achieved by setting $I^{(t1)} = I^{(t)} \cup \{a_1\}$, $X^{(t1)} = X^{(t)} \cup N_{C^{(t)}}(a_1)$, $F^{(t1)} = F^{(t)} \setminus (N_{C^{(t)}}(a_1) \cup \{a_1\})$ and, $I^{(t2)} = I^{(t)}$, $X^{(t2)} = X^{(t)} \cup \{a_1\}$, and $F^{(t2)} = F^{(t)} \setminus \{a_1\}$.

Figure 6.3 illustrates conflicting arc branching for the sample network depicted in Figure 4.1 and conflicting arc pair $(1, 3)$ and $(4, 5)$. Arc $(1, 3)$ is selected as the branching arc to be included in the first sub-problem in Figure 6.3a, which causes the deletion of arcs $(3, 5)$, $(4, 3)$ and $(4, 5)$. However, the sub-problem of the second branch depicted in Figure 6.3b is created by deleting arc $(1, 3)$ without including any arc.

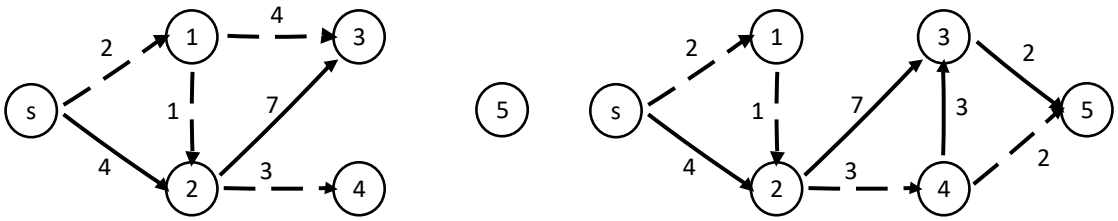


Figure 6.3. Conflicting arc branching: (a) First branch, (b) second branch.

6.2.4. Efficiency of the Branching Rules

The following proposition provides an idea on the potential total reductions in sub-problem sizes caused by the three branching rules. For this purpose we define $\epsilon_R^{(t)}$ as the amount of decrease in the number of free arcs after branching at node t of the BB search tree according to rule R. Here, P, K and A denote conflicting pair, clique and conflicting arc branching rules respectively for branching rule $R \in \{P, K, A\}$.

Proposition 6.1. $\epsilon_P^{(t)} \leq \epsilon_K^{(t)} \leq \epsilon_A^{(t)}$.

Proof. $\epsilon_P^{(t)} = 2$ since pair branching results in the deletion of one free arc per branch. Notice that if clique K is selected so that it only consists of a conflicting pair, then clique branching becomes equivalent to pair branching. Suppose K is a maximal clique and thus $\epsilon_K^{(t)} = |K| \geq 2$. Hence, the total decrease in the parent's free arcs can be larger. As for the conflicting arc branching, the total decrease in the parent's free arcs is $\epsilon_A^{(t)} = d_{C^{(t)}}(a_1) + 2$ assuming that a_1 is the branching arc. For any clique K of $C^{(t)}$ that includes a_1, $|K| \leq d_{C^{(t)}}(a_1) < \epsilon_A^{(t)}$. Here, a_1 is included and $N_{C^{(t)}}(a_1)$ are excluded in one branch and a_1 is excluded in the other, which results in a reduction of $(d_{C^{(t)}}(a_1) + 1) + 1$ arcs from the free arc set of the parent. □

As a result of Proposition 6.1, one can expect that the most efficient version of the conflict based BB algorithm could be obtained with the conflicting arc branching rule. Also, the division caused by conflicting arc branching is a partitioning and creates two disjoint subsets of parent's feasible solution set. Therefore, conflicting arc branching rule is implemented for the computational experiments. Its major difference is the following: It forces one of the arcs of the selected conflicting pair to be in a conflict free SPT, in one of the sub-problem's solution. However, the other two rules force arcs to be out of the feasible solutions by deleting them from the respective sub-networks.

6.3. Branching Variable Selection

When the relaxed sub-problem $\text{SP}^{(t)}$ is solved at node t of the BB search tree, obtained SPT can include conflicting arcs as discussed in Section 6.2. Hence, the selection of a conflicting arc pair from this SPT may greatly affect the performance of the BB algorithm. We apply a simple strategy, and try to find the first conflicting pair while searching the shortest path tree in breadth first manner starting at the root vertex s. We choose one of the conflicting arcs as the branching arc arbitrarily. Some other strategies, such as assigning to each arc a score equal to the number of conflicts in which it is involved inside the solution of the relaxation $\text{SP}^{(t)}$, and branching on the pair that contains the arc with the maximum score, are among the possibilities, which are beyond the scope of this thesis.

6.4. Sub-problem Selection

Sub-problem to process is selected with the smallest lower bound first (SLB) rule. SLB gives the highest selection priority to the sub-problem with the smallest lower bound value. In fact, this is potentially the biggest sub-problem that is to be divided. Hence, SLB is considered in the implementation of active node list of the new algorithm. The approach for bookkeeping is also very efficient due to a binary heap representing the active node list $\mathcal{L}$, organized according to non-decreasing LB.

6.5. Solution of the Relaxed Sub-problem

While branching at search node t, the two sub-problems of $\mathrm{SPC}^{(t)}$ are generated by means of two branching actions: the addition of a free arc to $X^{(t)}$, which forms the *exclusion* branch, and the addition of a free arc to $I^{(t)}$, which forms *inclusion* branch. Then, the relaxed optimal solutions of the two sub-problems, i.e., the two shortest path trees, can be obtained according to these two actions, from the shortest path tree $T^{(t)} = (V(T^{(t)}), A(T^{(t)}))$, which is an optimal solution of the parent's relaxed sub-problem $\mathrm{SP}^{(t)}$ and includes the arc subset $I^{(t)}$, i.e., $I^{(t)} \subseteq A(T^{(t)})$.

6.5.1. Exclusion Branch

Let $a = (p, q) \in F^{(t)}$ be an arc to be added to $X^{(t)}$, which means the deletion of a from $A(N^{(t)})$. If it is not an arc of the shortest path tree $T^{(t)} = (V(T^{(t)}), A(T^{(t)}))$, then the tree remains the same and a is deleted from $A(N^{(t)})$. However, if $a \in A(T^{(t)})$, then two subtrees are created after its deletion, resulting in primal infeasibility with respect to $T^{(t)}$.

6.5.2. Inclusion Branch

Let $a = (k, l) \in F^{(t)}$ be the arc to be added to $I^{(t)}$. According to arc branching rule it must also be an arc of the shortest path tree $T^{(t)} = (V(T^{(t)}), A(T^{(t)}))$, i.e.,

$a \in F^{(t)} \cap A(T^{(t)})$, and it must conflict with at least one other free arc of the tree. Notice that, all arcs conflicting with the arcs of $I^{(t)}$ have been previously deleted and a is still in $F^{(t)}$. There is an advantage of considering only deletions: we do not have to keep a list of arcs that must be in the SPT, i.e., included arcs $I^{(t)}$, at search node t. When an arc is included, any other inarc of its head vertex must be excluded, or equivalently if all other inarcs of the head vertex are excluded then, that arc must be included in the SPT. Hence, it is enough to keep only the list $X^{(t)}$ of the excluded arcs.

Thus, the addition of a to $I^{(t)}$ is valid with respect to arcs in $I^{(t)}$. After, the addition of a to $I^{(t)}$, all free arcs which are in conflict with a are deleted from subnetwork $N^{(t)}$, i.e., arcs $N_{C^{(t)}}(a) = N_C(a) \cap F^{(t)}$ are added to $X^{(t)}$. In other words, the inclusion of an arc a in branching is essentially equivalent to the exclusion of its conflicting arcs $N_{C^{(t)}}(a)$, and inarcs of its head vertex, due to the special structure of a SPT.

6.5.3. An Illustrative Example

The list of arcs to be deleted from a relaxed SP solution will increase either due to an arc being put directly to the list as it is excluded due to branching or due to an arc entering the included list potentially causing multiple number of its conflicted arcs to be removed from the relaxed solution. Therefore, the primal-dual algorithm (Figure 5.2) detailed at Chapter 5, can be applied for each deleted arc in order to efficiently reach the new relaxed solution increasing the lower bound in the process. Note that due to the SLB rule, global lower bound is increased as SPT arcs are deleted.

As an illustration of the conflicting arc branching rule, consider the sample network with given arc costs and conflict graph provided in Figure 6.4a and b respectively, at the initial search node of the BB search tree. Hence, $t = 0$, $I^{(0)} \equiv X^{(0)} \equiv \emptyset$ and $F^{(0)} \equiv A(N)$. An optimal spanning outtree of the relaxed problem $\text{SP}^{(0)}$, namely a shortest path tree, is given in Figure 6.4c with dashed arcs, where the numbers on arcs represent the reduced costs. It is determined using Dijkstra's Algorithm [59]. Say arc

$(0,1)$ is selected as the branching arc and sub-problem 1 is generated by adding it to $I^{(0)}$. Meanwhile, arc $(1,4)$ is in conflict with it and must be deleted from $A(N)$ and added to $X^{(0)}$. Notice that arc $(3,1)$ must also be deleted since it is also an inarc of vertex 1. This results in the disconnected tree of sub-problem 1 given in Figure 6.4d with $S = \{0,1,2,3\}$ and $\overline{S} = \{4,5\}$. Arcs $(3,5)$ and $(2,4)$ are from S to $\overline{S}$ and have the smallest reduced cost 2. Any one of them can be selected as new tree arcs connecting $T_1^{(0)}$ and $T_2^{(0)}$. Observe that if arc $(3,5)$ is selected, then $(4,5)$ is forced to leave since it is also an inarc of vertex 5.

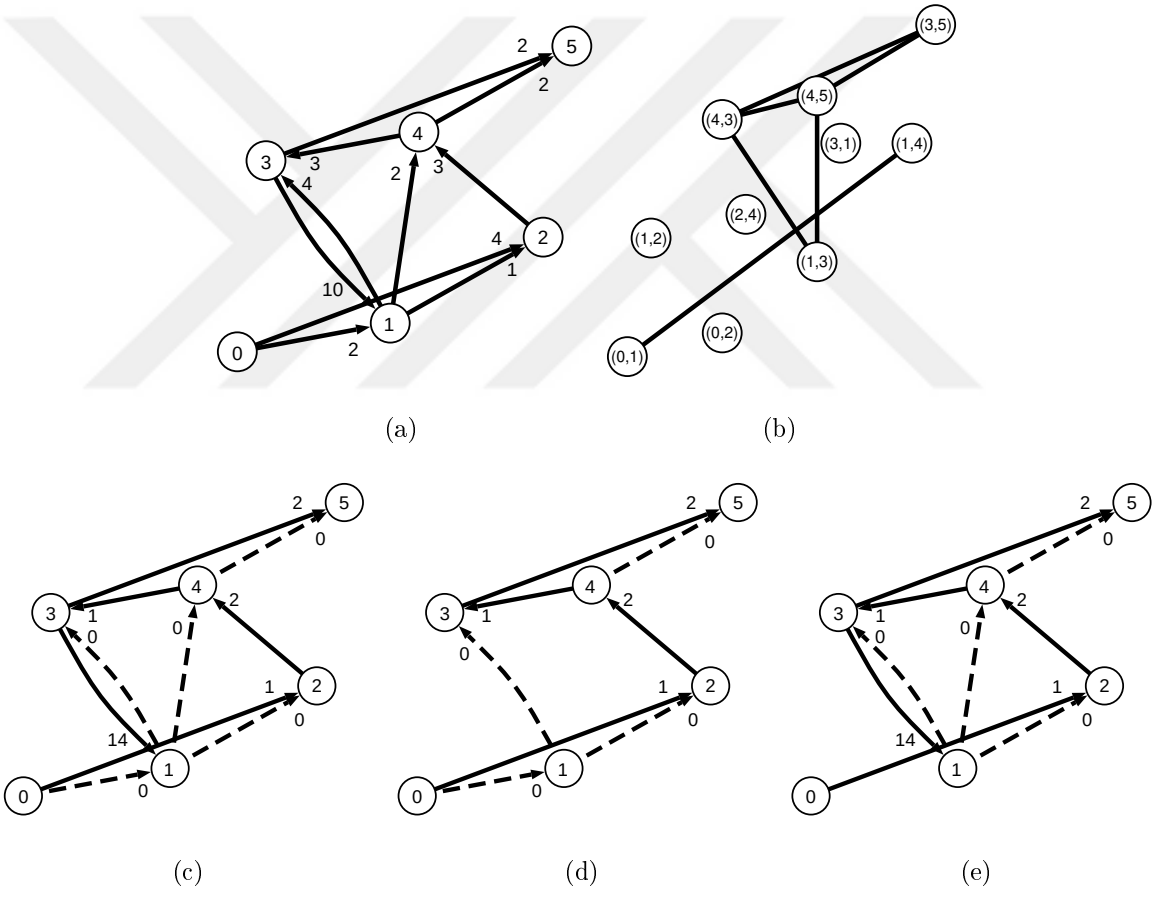


Figure 6.4. (a) Sample network, (b) sample conflict graph, (c) initial SPT, (d) sub-problem 1, (e) sub-problem 2

Second branch is more interesting as we will illustrate the restoration algorithm on it. Here arc $(0,1)$ is deleted from $A(N)$ by adding it to $X^{(0)}$ generating sub-problem 2 whose disconnected tree is depicted with Figure 6.4e with $S = \{0\}$ and $\overline{S} = \{1,2,3,4,5\}$. Arc $(0,2)$ is the only arc from S to $\overline{S}$ and has the reduced cost of 1. Hence the entering arc (k,l) is selected to be $(0,2)$ and leaving arc becomes

$(p, l) = (1, 2)$ with $S^{new} = \{2\}$. Dual simplex steps have not been completed yet; reduced cost must be updated according to the update scheme in the primal-dual algorithm (Figure 5.2) and arc exchanges, i.e., pivot operations, will be repeated after S^{new} is determined until the reduced costs become non-negative in the reoptimized tree or infeasibility is detected. Illustration is continued for the second branch in Figure 6.5 where dashed and solid arcs belong to the tree and non-tree arcs respectively.

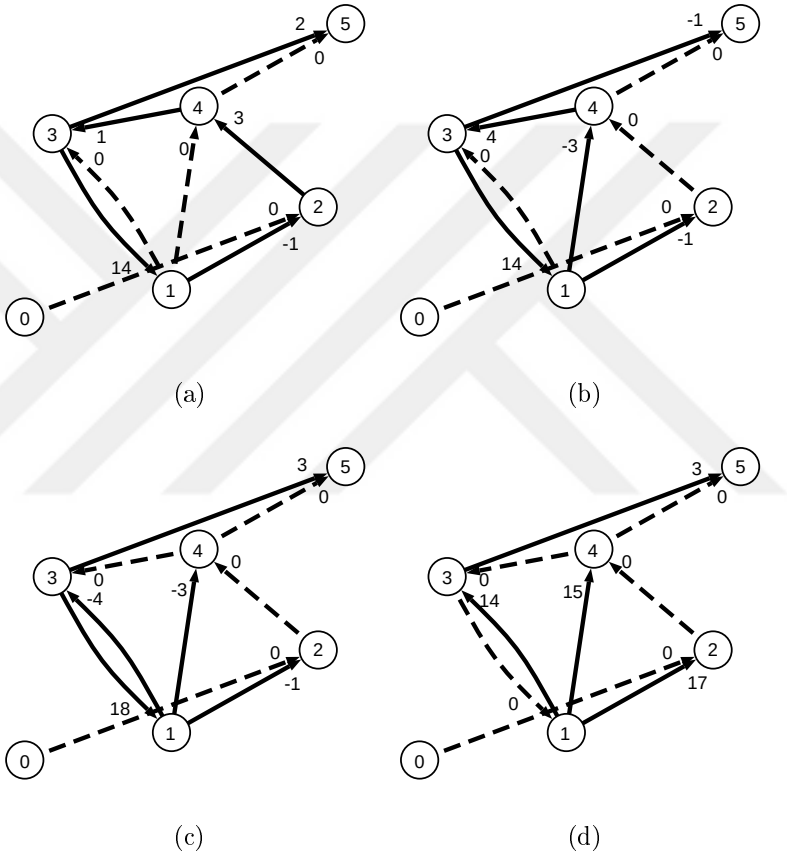


Figure 6.5. Reoptimization of sub-problem 2: (a) Iteration 1, (b) iteration 2, (c) iteration 3, (d) re-optimized SPT of sub-problem 2.

In the end of the first iteration (as arc $(0, 2)$ is inserted), the picture looks like Figure 6.5a with $S = \{0, 2\}$ and $\overline{S} = \{1, 3, 4, 5\}$. The only arc from S to $\overline{S}$ is $(2, 4)$ with reduced cost 3. Hence the entering arc (k, l) is selected to be $(2, 4)$ and leaving arc becomes $(p, l) = (1, 4)$ with $S^{new} = \{4, 5\}$ as the subtree rooted at l is moved into T_1. Resulting picture at the end of the second iteration can be seen at Figure 6.5b. Here $S = \{0, 2, 4, 5\}$ and $\overline{S} = \{1, 3\}$. Arc $(4, 3)$ is from S to $\overline{S}$ and has the smallest reduced cost of 4. Hence the entering arc (k, l) is selected to be $(4, 3)$ and leaving arc

becomes $(p, l) = (1, 3)$ with $S^{new} = \{3\}$. New picture at the end of the third iteration can be seen at Figure 6.5c. As the last iteration with $S = \{0, 2, 3, 4, 5\}$ and $\overline{S} = \{1\}$, only vertex 1 is out of the span of partial tree T_1. Arc $(3, 1)$ is the sole arc from S to $\overline{S}$ with reduced cost 18 it has to be selected. Hence the entering arc (k, l) is selected to be $(3, 1)$ with $S^{new} = \{1\}$ and as the last iteration we do not have a leaving arc. The resulting reoptimized SPT, which is the relaxed solution of sub-problem 2, is given in Figure 6.5d. When the new shortest path tree is obtained, there is no arcs with negative reduced cost, meaning that the tree is an optimal solution of the relaxation of sub-problem 2. Besides, except source vertex s, all vertices have exactly one inarc as required for a SPT, and the arc set is $\{(0, 2), (2, 4), (4, 3), (3, 1), (4, 5)\}$.

6.6. Infeasibility Test

Infeasibility test procedure is based on two simple observations:

(i) If there is a free arc $a \in F^{(t)}$ that is the only inarc of a vertex $v \in V(N) \setminus \{s\}$, then it must be included in the SPT, since it is not possible to construct a SPT rooted at the source node s, having a directed s, v-path without a;

(ii) If there is a vertex $v \in V(N)$ with no inarc from the arc subset $I^{(t)} \cup F^{(t)}$ then sub-problem SPC$^{(t)}$ is infeasible since it is not possible to construct a SPT including the arcs in $I^{(t)}$ and rooted at the source node s, having a directed s, v-path.

In other words, if there is a vertex with single inarc a, then it is added to $I^{(t)}$ and all arcs conflicting with a, i.e., $N_{C^{(t)}}(a)$, are deleted. In fact, it is possible to repeat these two steps alternately until one of the three outputs given below is reached when it is run on sub-problem SPC$^{(t)}$:

(i) The infeasibility of SPC$^{(t)}$ is detected;

(ii) An optimal conflict free shortest path tree that includes the arcs in $I^{(t)}$ is found;

(iii) No decision can be made about SPC$^{(t)}$, i.e., none of the above two cases occur.

Infeasibility test is formally given in Figure 6.6 below. $N^{(t)} = (V(N^{(t)}), A(N^{(t)}))$ and $C^{(t)} = (V(C^{(t)}), E(C^{(t)}))$ are respectively the network and conflict graph associated with sub-problem SPC$^{(t)}$ where $V(N^{(t)}) \equiv V(N)$, $A(N^{(t)}) \equiv I^{(t)} \cup F^{(t)}$, $V(C^{(t)}) \equiv A(N^{(t)})$, and $E(C^{(t)}) \equiv F^{(t)}$ hold as previously mentioned.

Input: Sub-problem SPC$^{(t)}$ represented with $A(N^{(t)}) = I^{(t)} \cup F^{(t)}$ and $X^{(t)}$
Output: **true** if SPC$^{(t)}$ is infeasible, **false** otherwise
begin
list $\leftarrow \emptyset$ ▷ *Temporary vertices to visit with in-degree 1*
for all $v \in V(N^{(t)})$ **do**
  if $v \neq s$ **and** $d^-_{N^{(t)}}(v) = 0$ **then**
    return true ▷ *Infeasible with an unreachable vertex*
  else if $v \neq s$ **and** $d^-_{N^{(t)}}(v) = 1$ **then**
    list $\leftarrow$ list $\cup \{v\}$ ▷ *Included at step 10*
while list **not** $\emptyset$ **do** ▷ *For each vertex with in-degree 1*
  Select a vertex v in list
  list $\leftarrow$ list $\setminus \{v\}$
  $I^{(t)} \leftarrow I^{(t)} \bigcup \{a\}$ **and** $F^{(t)} \leftarrow F^{(t)} \setminus \{a\}$ ▷ *$a = (u, v)$ be the only arc to v*
  for all $(y, z) \in N_{C^{(t)}}(a)$ **do**
    $F^{(t)} \leftarrow F^{(t)} \setminus \{(y, z)\}$ **and** $X^{(t)} \leftarrow X^{(t)} \bigcup \{(y, z)\}$ ▷ *Remove conflicting*
    if $z \neq s$ **and** $d^-_{N^{(t)}}(z) = 0$ **then**
      return true ▷ *Infeasible with an unreachable vertex*
    else if $z \neq s$ **and** $d^-_{N^{(t)}}(z) = 1$ **then**
      list $\leftarrow$ list $\cup \{z\}$ ▷ *Included at step 10*
return false ▷ *Unable to detect infeasibility, problem size is reduced*
end

Figure 6.6. Infeasibility test.

Notice that, if a spanning outtree is constructed as a result of the alternation, then it is a conflict free SPT, since every vertex, except s, has single inarc, only inarcs that must be on a SPT are added and they are all conflict free. Unfortunately, this case, resulting in a feasible solution of SPC is extremely rare and we skip this control in

the algorithm. In fact, this can be detected by checking whether $|I^{(t)}| = |V(N)| - 1$ at the very end or when the relaxed solution is conflict free. Hence, two possible outputs of our implementation are the infeasibility of $\mathrm{SPC}^{(t)}$ and no decision about its status, which occurs when the alternation ends up with the situation where each vertex has at least two inarcs. The infeasibility decision is based on the existence of a vertex having no inarcs. Clearly, this is a sufficient condition, implying the possibility of an infeasible sub-problem even if such vertex does not exist.

There can be several cases for undetected infeasibility. For example, the connectedness of network N is not guaranteed after the infeasibility test is applied. In other words, guaranteeing an incoming arc for each vertex does not guarantee a connected sub-network. One can argue that an additional connectedness control can be made. First, there is a trade-off between computational time and performance, and second even that does not guarantee feasibility. It is possible to give a counter example for infeasibility in the second case. Consider two bridges which are in conflict with each other in the sub-network, where a bridge is such an arc that makes the network disconnected without it. In this connected graph, a feasible solution requires both bridges on the SPT, which is not possible due to the conflict relation between two bridges. There is no conflict free feasible solution and this problem is infeasible.

It is also possible to check the infeasibility of $\mathrm{SPC}^{(t)}$ by solving its relaxation $\mathrm{SP}^{(t)}$, but this can be computationally more expensive than running the infeasibility test procedure. The whole purpose is the detection of the infeasibility of $\mathrm{SPC}^{(t)}$ without necessarily solving $\mathrm{SP}^{(t)}$, at a given BB search node t.

The infeasibility test procedure is applied to the two sub-problems $\mathrm{SPC}^{(ti)}$, $i = 1, 2$, right after the division of $\mathrm{SPC}^{(t)}$ at parent search node t, in order to check whether they are worth adding to the active node list $\mathcal{L}$. As it can be seen from the results of the computational experiments presented in Chapter 9, in addition to its simplicity, it is fast and considerably reduces the size of the sub-problems, contributing to the overall efficiency of the BB algorithm.

6.7. Conflict Based Algorithm

The algorithm starts by solving one-to-many shortest path problem, i.e., $SP^{(0)}$, as the initial relaxation and generate the initial SPT. This can be done using one of the known algorithms. Relaxation from scratch is only solved at initial search node once, therefore its efficiency does not matter for the overall algorithm. We use the heap implementation of Dijkstra's algorithm [59]. At each iteration, the sub-problem with the smallest relaxation value (SLB) is selected as the current active search node, and the active node list is cleaned by removing the sub-problems with lower bounds larger than the current upper bound (i.e., the value of the incumbent) from the active node list. If the SPT of the selected sub-problem is conflict free and its objective value is less than the current upper bound, then the incumbent and the upper bound are updated. Otherwise, the SPT has conflicting arcs, a pair of them is selected and new sub-problems are created according to the arc branching rule.

Newly generated sub-problems are first filtered by means of the infeasibility test (Figure 6.6). The purpose is to detect their infeasibility, using computationally efficient sufficient conditions. Obviously, these conditions can miss infeasible sub-problems when the test is indecisive, which can be captured when their relaxations are solved. The procedure still helps the efficiency by reducing the number of arcs. This increases the efficiency of the reoptimization of the parent's SPT considerably, which is realized for computing the newly created child sub-problem's relaxation bounds. Infeasibility test is utilized before the relaxed objective value for the sub-problem is calculated by primal-dual (Figure 5.2). Calculated SPT is a solution of SPC if it is conflict free. The sub-problem is added to active problem list, which is maintained as a binary heap. The algorithm terminates when the active node list is empty or the time limit is reached. The pseudo-code for the proposed BB based on conflicts utilizing arc branching rule (BBA) is given in Figure 6.7.

Input: A SPC problem input $N = (V(N), A(N)))$, $c_a \geq 0$ for $a \in A(N)$, and conflict graph $C = (V(C), A(C))$ where $V(C) \equiv A(N)$

Output: Best found solution within time limit

begin

Create initial search node 0, $\bar{z} \leftarrow \infty$ ▷ *Initialization*

$\mathcal{L} \leftarrow \{0\}$ ▷ *Active node list, $\mathcal{L}$, is a non decreasing heap of relaxation cost*

while $|\mathcal{L}| > 0$ **and** time limit is **not** reached **do** ▷ *Termination test*

  $\theta \leftarrow$ Sub-problem with smallest LB ▷ *Problem selection*

  if LB(θ) $< \bar{z}$ **then** ▷ *Cleaning at runtime*

    if There exists conflicted pair of arcs **then** ▷ *Detection*

      Create sub-problems ξ_1, ξ_2 with the arc branching rule ▷ *Division*

      for all Sub-problem $\xi \in \{\xi_1, \xi_2\}$ **do**

        if not InfeasTest(ξ) **then** ▷ *Pruning*

          Re-optimize SPT of θ to get SPT of ξ ▷ *LB calculation*

          if SPT of ξ exists **then**

            $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{\xi\}$ ▷ *Add sub-problem to $\mathcal{L}$*

    else

      $\bar{z} \leftarrow$ LB(θ) ▷ *Conflict free feasible solution*

  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\theta\}$ ▷ *Sub-problem θ has been processed*

return $\bar{z}$ **and** *corresponding feasible solution* **as the best feasible solution**

end

Figure 6.7. BBA: Branch-and-bound based on conflicts using arc branching.

7. DIVING

A depth first search diving algorithm with an emphasis on finding a feasible solution as fast as possible is developed in this chapter. While it can be used at any node of BB search tree in order to catch a feasible solution, when it is run on the root node, it behaves as if it were a complete solution algorithm itself. It systematically generates all possible solutions, and calculates the objective function only if it finds a feasible solution.

Diving gives emphasis on fast enumeration of the generated search nodes rather than the reduction of the solution space. Intelligent pruning techniques used by BB, i.e., lower bounds and the infeasibility test aiming the reduction of search space, are not considered. Hence, memory requirement of diving can be exhaustive compared to other algorithms for SPC.

7.1. Branching Strategy

The branching strategy for diving is inherited from outtree property of SPT. As a necessary condition for outtree structure, each vertex except source s has to have a single incoming arc. Using this idea, diving tries to assign each vertex one inarc to cover it. First, an uncovered vertex is found with no incoming arc, and first available arc is assigned to it in order to facilitate branching. Hence, one arc is set to included to generate a sub-problem, similar to the arc branching used in BBA (Section 6.2.3). Parent sub-problem is then utilized as the second branch where the same arc is set to excluded. The difference between the two branching rules is as follows: arc branching is utilized in BBA in order to break an infeasibility generating conflict pair in the relaxed solution. In diving, branching is used to generate sub-problems without information about conflict relations or the relaxed solution.

7.2. Circuit Prevention

One of the important factors when designing such an algorithm for SPC is to deal with potential circuits that might be generated. Recall that, the feasible solutions for the BBA explained in Chapter 6 are obtained from the reoptimization algorithm and a SPT is by definition circuit free. However, solving the relaxation at each iteration is time consuming. As explained in Section 6.1, branching may create circuits on the solution without using the SPT. Hence, an approach is needed in order to prevent circuits in potential feasible solutions.

There are two main approaches to overcome potential circuits in a solution generated by diving. First approach is to forbid circuits at intermediate steps of the algorithm. As a potential method for the first approach; start from root vertex s and grow the tree one arc at a time, guaranteeing a feasible solution at the end without circuits. This is costly in inclusion step since a check is made to add an arc that grows the tree. Another potential method for this approach may be to keep growing multiple trees simultaneously (a forest of outtrees, see Section 8.2) and checking for circuits at each inclusion as multiple outtrees merge throughout the iterations. This method also has overhead costs associated with merging of trees.

Second approach is to allow circuits while generating solutions and check for them after a potential solution is generated, which is adopted in our diving implementation. Incumbent is updated if there is no circuit in a potential solution, which is a subset of arcs with size $|V(N)| - 1$.

7.3. Search Strategy

Whenever a new arc a is to be included using the branching strategy explained in Section 7.1, a's conflicted arcs, $N_C(a)$, are removed from its sub-problem. Then, this new sub-problem is added to $\mathcal{L}$, which is represented as a stack, facilitating depth-first search. However, we can not simply delete the parent problem because it needs

to stay in the active node list $\mathcal{L}$ in order to accommodate full search of the solution space. The parent problem simply becomes the second sub-problem with the removal of the branching arc a. When $\mathcal{L}$ is empty, all possible solutions are enumerated. The pseudo-code of the algorithm is given in Figure 7.1.

Input: A SPC problem input $N = (V(N), A(N)))$, $c_a \geq 0$ for $a \in A(N)$, and conflict graph $C = (V(C), A(C))$ where $V(C) \equiv A(N)$

Output: Best found solution within time limit

begin

1: Create initial search node 0, $\bar{z} \leftarrow \infty$ ▷ *Initialization*
2: $\mathcal{L} \leftarrow \{0\}$ ▷ *Active node stack*
3: **while** $|\mathcal{L}| > 0$ **and** time limit is not reached **do** ▷ *Termination test*
4: $\theta \leftarrow$ last element of $\mathcal{L}$ ▷ *Problem selection*
5: **if** $|I^{(\theta)}| = |V(N)| - 1$ **then** ▷ $I^{(\theta)}$ *is a candidate feasible solution*
6: **if** $I^{(\theta)}$ circuit free **then** ▷ $I^{(\theta)}$ *is a feasible solution*
7: Calculate cost of the tree represented by $I^{(\theta)}$ as $\text{Cost}(\theta)$
8: **if** $\text{Cost}(\theta) < \bar{z}$ **then**
9: $\bar{z} \leftarrow \text{Cost}(\theta)$ ▷ *Conflict free feasible solution*
10: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\theta\}$ ▷ *Sub-problem at search node θ has been processed*
11: **go to** 3
12: **for all** $v \in V(N)$ **do**
13: **if** $\nexists u : (u, v) \in I^{(\theta)}$ **then** ▷ *Vertex v does not have selected inarc*
14: Create sub-problem ξ arbitrarily selecting an inarc (u, v) ▷ *Division*
15: $I^{(\xi)} \leftarrow I^{(\theta)} \bigcup \{(u, v)\}, N^{(\xi)} \leftarrow N^{(\theta)} \setminus \{(i, v) : (i, v) \in A(N^{(\theta)}), i \neq u\}$
16: $N^{(\xi)} \leftarrow N^{(\xi)} \setminus N_C((u, v))$ ▷ *Conflict clear for branch arc*
17: $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{\xi\}, N^{(\theta)} \leftarrow N^{(\theta)} \setminus \{(u, v)\}$ ▷ *θ stays in $\mathcal{L}$ as second branch*
18: **go to** 3
19: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\theta\}$ ▷ *Not able to find a suitable arc to set to one, θ is infeasible*
20: **return** $\bar{z}$ **and** *corresponding feasible solution* **as the best feasible solution**

end

Figure 7.1. Diving algorithm.

7.4. An Extended Diving Algorithm

A variant of diving is developed in order to reduce memory requirement of the algorithm presented in Figure 7.1. The algorithm follows a predetermined sequence O of arcs. This sequence of arcs is set at the initialization step of the algorithm and remains unchanged until the algorithm terminates. In the sequence O, arcs are set to one and conflicting arcs are removed until either a feasible solution is found or infeasibility is detected. When one of these two cases occurs, we perform a backtrack operation on the current status of the algorithm in order to continue enumerating the remaining solution space.

Whenever the number of remaining free arcs are not enough to complete the tree, this sub-problem is infeasible. In addition to this control, infeasibility of the current sub-problem is detected using the infeasibility test in Figure 6.6. In either case, backtracking should be completed in order to return the current solution into a state prior to the last included arc. Then, this included arc is considered as excluded and the algorithm continues until entire solution space is searched or time limit is reached.

The approach is different from complete enumeration because of the infeasibility test and circuit prevention. The hardest part of the algorithm is to implement the required data structure in order to be able to facilitate backtracking on the current partial solution. This extended diving algorithm is given in Figure 7.2.

Unfortunately, extended diving does not perform well. Maintaining the data structures needed for infeasibility test and circuit prevention techniques cause too much overhead and do not justify the memory improvement at the end (see Section 9.1.5). Due to the lack of performance improvement, all computations are conducted using the original algorithm given in Figure 7.1.

Input: A SPC problem input $N = (V(N), A(N)))$, $c_a \geq 0$ for $a \in A(N)$, and conflict graph $C = (V(C), A(C))$ where $V(C) \equiv A(N)$

Output: Best found solution within time limit

begin

1: Build ordering O of the arcs in non-decreasing degrees in C ▷ *Initialization*

2: $t \leftarrow 0$ ▷ *Iteration counter*

3: **while** Timelimit is not reached **do** ▷ *Termination test*

4: **if** $|I^{(t)}| = |V(N)| - 1$ **then**

5: Calculate feasible solution cost, update $\bar{z}$ if necessary

6: **else**

7: Perform Infeasibility Test on $N^{(t)} = I^{(t)} \bigcup F^{(t)}$

8: **if** $|I^{(t)}| = |V(N)| - 1$ **or** Infeasible **or** $|I^{(t)} \bigcup F^{(t)}| < |V(N)| - 1$ **then**

9: Restore $I^{(t+1)}, X^{(t+1)}$ from $I^{(t)}, X^{(t)}$ ▷ *Backtrack reverting Step 14 and 15*

10: $X^{(t+1)} \leftarrow X^{(t+1)} \setminus \{a\}$ ▷ *Where a is the most recent branching arc*

11: $t \leftarrow t + 1$

12: **go to** 3

13: Find the first free arc in ordering $a \in O \bigcap F^{(t)}$ where $I^{(t)} \bigcup \{a\}$ circuit free

14: $I^{(t+1)} \leftarrow I^{(t)} \bigcup \{a\}$ ▷ *Branching*

15: $X^{(t+1)} \leftarrow X^{(t)} \bigcup N_C(a)$

16: $t \leftarrow t + 1$

17: **return** $\bar{z}$ **and** *corresponding feasible solution* **as the best feasible solution**

end

Figure 7.2. Extended diving algorithm.

8. A BRANCH AND BOUND ALGORITHM BASED ON STABLE SETS

A feasible solution of SPC is a conflict free spanning outtree rooted at s on network N. BBA, which is given in Chapter 6, benefits from the information provided by the SP relaxations and cleans conflicting arc pairs that exist in a SPT using branching rules based on these conflicting arc pairs. Hence, at a node of the BB search tree, the aim is to transform a SPT of N into a stable set of C.

BB based on stable sets (BBSS), which is introduced in this chapter, benefits from the information provided by the (vertex) maximal stable sets of C and tries to improve them using branching rules based on their size or cost deficiencies. Hence, at a node of the BB search tree, the aim is to transform a maximal stable set of C into a SPT of N.

8.1. Branching Rules

BB search tree generated by the stable set based branching does not have to be a binary tree in contrast with the branching rules explained for BBA in Section 6.2. This is because the feasible solution set of a sub-problem can be divided in more than two branches resulting in the generation of more than two new sub-problems.

As can be remembered, at search node t, when a new free arc $a \in F^{(t)}$ is set to included as the set becomes $I^{(t)} \cup \{a\}$, the subset $N_{C^{(t)}}(a)$ of free arcs conflicting with it are deleted from $F^{(t)}$, which can clearly cause considerable reduction in the size of newly generated sub-problems. In fact, this is the main motivation behind this branching rule adopted from the early works [61, 62] and [63] respectively on the *Maximum Clique* (MC) and *Maximum Weight Stable Set* (MWS) problems.

8.1.1. Extended Conflict Graph

Since a feasible solution of SPC is a spanning outtree, every vertex beside root s, which is not allowed to have inarcs, must have exactly one inarc by definition. Hence, the inarcs of vertices $v \in V(N) \setminus \{s\}$ are in conflict with each other by definition. As a result one can extend conflict graph C by adding these conflict relations as new edges. The extended conflict graph is denoted as $\widehat{C} = (V(\widehat{C}), E(\widehat{C}))$ where $V(\widehat{C}) \equiv V(C) \equiv A(N)$ and $E(\widehat{C}) = E(C) \cup (\cup_{v \in V(N) \setminus \{s\}} \delta_N^-(v))$.

Recall that a conflict free spanning outtree of network N, which is a feasible solution of SPC, has the following properties:

(i) It is a stable set of conflict graph C,
(ii) The size of the stable set is $|V(N)| - 1$,
(iii) The elements of the stable set, which are the arcs of N, do not contain a circuit,
(iv) Each element of the stable set is an inarc to a different vertex of N, except s.

Clearly for a stable set satisfying property (ii), the last property is satisfied automatically on the extended conflict graph $\widehat{C}$ because of the newly added conflict relations. In other words, every feasible solution of SPC becomes a circuit free stable set of extended conflict graph $\widehat{C}$ with size $|V(N)| - 1$, i.e., every feasible solution of SPC must satisfy all four properties.

As a consequence of this discussion, it is possible to see that the cardinality of a circuit free stable set of $\widehat{C}$ cannot be larger than $|V(N)| - 1$, and an optimal solution of SPC is one of the stable sets of $\widehat{C}$ with size $|V(N)| - 1$ that is also a SPT of N. Therefore, the stable set based branching rule tries to make up size deficiency first, while paying attention to the circuit freeness until a feasible solution of SPC is obtained. This is feasibility branching. Once a feasible solution is obtained, the rule tries to transform it into a SPT, trying to make it the shortest tree, in order to obtain an optimal solution of SPC by means of optimality branching.

8.1.2. Optimality Branching

At node t of the BB search tree the arcs in the included set $I^{(t)}$ is a stable set of conflict graph $\widehat{C}$. Let $\mathscr{S}^{(t)}$ be a stable set of conflict subgraph $\widehat{C}^{(t)} = \widehat{C}[F^{(t)}]$. Then, $I^{(t)} \cup \mathscr{S}^{(t)}$ is also a stable set of $\widehat{C}$ by definition. Besides, if $I^{(t)} \cup \mathscr{S}^{(t)}$ is circuit free, and $|\mathscr{S}^{(t)}| = (|V(N)| - 1) - |I^{(t)}|$, the arcs in $I^{(t)} \cup \mathscr{S}^{(t)}$ form a spanning outtree of N rooted at s. It is also a SPT of a subnetwork of N maximal with respect to this property. Notice that this maximal subnetwork spans N since its vertex set is $V(N)$. Hence, any other SPT with smaller total shortest distances must have at least one arc from the outside the arc set of the maximal subnetwork and all of such arcs constitutes the branching vertices of conflict subgraph $\widehat{C}^{(t)}$.

Clearly, when one-to-many shortest path problem is solved on the aforementioned maximal subnetwork, an optimal solution is the SPT with arcs $I^{(t)} \cup \mathscr{S}^{(t)}$. Hence, the reduced costs of the arcs of the maximal subnetwork, calculated with the dual solution optimal relative to $I^{(t)} \cup \mathscr{S}^{(t)}$, are nonnegative as a consequence of the fact that SP is essentially an LP, and its solution on the maximal subnetwork give $I^{(t)} \cup \mathscr{S}^{(t)}$ as an optimal solution. In short, the new branching elements are vertices of $\widehat{C}^{(t)}$ corresponding to arcs of N with negative reduced costs relative to $I^{(t)} \cup \mathscr{S}^{(t)}$. All of them can be added to $I^{(t)}$ one by one for generating new subproblems provided that the new included set is circuit free.

Consider a branching arc with negative reduced cost. If this arc is not in conflict with any arcs in $\mathscr{S}^{(t)}$, then it is possible to immediately do a local improvement without the need for branching. In other words, for a stable set of full size, i.e. $|\mathscr{S}^{(t)}| = (|V(N)| - 1) - |I^{(t)}|$, each arc with negative reduced cost is inserted into $\mathscr{S}^{(t)}$ one by one. If the size of $\mathscr{S}^{(t)}$ does not decrease after an insertion due to a conflict, a local improvement on the objective value is successfully completed. Then, branching is continued with the remaining negative reduced costs arcs.

8.1.3. Feasibility Branching

Prior to applying optimality branching, we have to solve a maximum stable set problem with additional circuit elimination constraints on conflict subgraph $\widehat{C}^{(t)}$, which is computationally demanding. Hence, we focus on (vertex) maximal stable sets of $\widehat{C}^{(t)}$ instead of maximum ones. Let $\mathscr{S}^{(t)}$ be a maximal stable set of $\widehat{C}^{(t)}$ such that $|\mathscr{S}^{(t)}| < (|V(N)| - 1) - |I^{(t)}|$. Then, a larger stable set must include at least one of the arcs in $F^{(t)} \setminus \mathscr{S}^{(t)}$. Hence, all of them can be added to $I^{(t)}$ for generating new subproblems provided that the new included set is circuit free.

Still, it is possible to economize in the number of branches by considering a sub-graph of $\widehat{C}^{(t)}$ which is maximum in vertex size and with respect to the property that it includes $\mathscr{S}^{(t)}$ as its maximum stable set. In other words, the stability number of this sub-graph is $|\mathscr{S}^{(t)}|$ and its vertex size is maximum possible with respect to this property. As can be realized this is another computationally demanding problem. However, a vertex maximal sub-graph can work out. One quick approach is to take the union of disjoint maximal cliques of $\widehat{C}^{(t)}$ each of which includes exactly one vertex of $\mathscr{S}^{(t)}$. Clearly, the stability number of this sub-graph is $|\mathscr{S}^{(t)}|$ and it is maximal. As a consequence, once a maximal stable set $\mathscr{S}^{(t)}$ of $\widehat{C}^{(t)}$ is determined, such a maximal sub-graph can be obtained by checking one by one the vertices in $F^{(t)} \setminus \mathscr{S}^{(t)}$ whether they from a clique with one of the vertices in $\mathscr{S}^{(t)}$. Then, the arcs inside such a maximal sub-graph called $W^{(t)}$, can not be considered as candidates for branching, because their insertion can not increase the size of the stable set $\mathscr{S}^{(t)}$. A maximal stable set can be determined using the greedy heuristic given in Figure 8.1.

The greedy stable set algorithm considers each vertex $a \in V(\widehat{C})$ $(\equiv A(N))$ in non-decreasing order of their degrees $d_{\widehat{C}}(a)$, i.e. $O = \{a^{(1)}, a^{(2)}, ..., a^{(|A(N)|)}\}$. Each vertex a is added to the stable set in order O, all neighbours of a, i.e. $N_{\widehat{C}}(a)$ is deleted, and algorithm continues with the next available vertex in the ordering O. This greedy approach aims to construct a stable set as large as possible by selecting lowest degree vertices first. Similarly, vertex maximal sub-graph $W^{(t)}$ can be obtained with a greedy

algorithm similar to Figure 8.1, checking each vertex whether they form a clique with one of the vertices of $\mathscr{S}^{(t)}$.

Input: An extended conflict graph $\widehat{C}$, corresponding network N

Output: A stable set $\mathscr{S}$

begin

1: **for** Each vertex $a \in V(\widehat{C})$ in non decreasing order of $d_{\widehat{C}}(a)$ **do**

2: **if** $\mathscr{S} \cup \{a\}$ does not imply a circuit in N **then**

3: $\mathscr{S} \leftarrow \mathscr{S} \cup \{a\}$

4: $\widehat{C} \leftarrow \widehat{C} \setminus (\{a\} \cup N_{\widehat{C}}(a))$

5: **return** $\mathscr{S}$

end

Figure 8.1. Greedy maximum cardinality stable set algorithm.

In [63] a branching convention is proposed in order to reduce the size of the overall BB tree. Let $\{a_t^{(1)}, a_t^{(2)}, ..., a_t^{(k)}\}$, give k arcs to be branched on one-by-one on a BB search node t, with respect to some ordering. When branching on arc $a_t^{(i)}$, $i \leq k$, all arcs in $\{a_t^{(1)}, a_t^{(2)}, ..., a_t^{(i-1)}\}$ are set to zero. This lexicographical rule eliminates the possible emergence of the same solution on different parts of the search tree. Note that any ordering works for the ordering of branch arcs. Since we already have the ordering O used in the stable set algorithm in Figure 8.1, same ordering is used for this purpose, both in optimality and feasibility branching.

8.2. Circuit Prevention

As explained in Section 6.1, feasible solutions for the BBA are obtained from the reoptimization algorithm and SPTs are by definition circuit free. However, similar to the discussion about circuits in diving (see Section 7.2), they need to be dealt with for BBSS. In diving implementation, circuits are allowed rather than prevented in order to process search nodes as fast as possible. However, for BBSS circuit prevention is easier to implement due to the availability of data structures developed and maintained in order to facilitate the algorithm. Our preliminary computational tests have shown

that circuit prevention works better for this BB implementation, compared to allowing circuits.

The implementation of circuit prevention feature works on the following principle: A circuit free stable set $\mathscr{S}$ of $\widehat{C}$ is a forest of outtrees, i.e., can be represented by a disjoint set data structure for each tree. Individual trees grow and merge as more arcs are inserted into $\mathscr{S}$. Circuit check is completed at each insertion into $\mathscr{S}$. If both vertices of an arc are in the same outtree, there is a circuit and inclusion of such arc into the set $\mathscr{S}$ is not valid. The necessary data structures and functions are implemented in order to facilitate the merging of different trees in the forest as valid arcs are inserted into $\mathscr{S}$. Stable set selection heuristic given in Figure 8.1 also uses this circuit preventing approach, making generated $\mathscr{S}$ circuit-free. Circuit generating branching arcs are also eliminated using the same approach in BBSS given in Figure 8.2

There is an interesting interaction between the maximal sub-graph W (see Section 8.1.3) and the type of approach taken while dealing with circuits in the greedy algorithm depicted with Figure 8.1. Compared to circuit detection approach which allows for circuits first and checks after, circuit prevention removes arcs from being a candidate to be in the maximal sub-graph W (which is obtained by clique extension of $\mathscr{S}$), which in turn reduces the number of branches. Hence, W is not unnecessarily populated with arcs that are irrelevant (i.e., creates a circuit inside $\mathscr{S}$). This improvement gives a 25% reduction of solution times in the preliminary results and potential circuits are automatically eliminated whenever an arc is selected for $\mathscr{S}$ or as a branching variable.

8.3. Infeasibility Test

A fast infeasibility test has a potential to significantly reduce the number of BB search nodes calculated, increasing the efficiency. Infeasibility test is explained in detail for BBA in Section 6.6. That infeasibility test primarily detects infeasibility by checking vertex in-degrees. However, for BBSS, we find it useful to add a quick *Breadth First Search* (BFS) connectivity test at the end of infeasibility test as a last step. In

other words, checking infeasibility with BFS is faster than continuing branching until it is proven by having no candidate arcs for feasibility branching (i.e., $F^{(t)} \setminus W^{(t)} = \emptyset$, at BB search node t).

8.4. Lower Bound Calculation

A lower bound is a key attribute of a BB search node. With a good lower bound and a good incumbent solution, the number of BB nodes to be processed can be reduced significantly. In BBSS, selecting branching variables does not use the information available in a SPT. Therefore, maintaining and re-optimizing a SPT does not prove useful for our implementation. Unlike BBA where branching arc is selected from SPT, arcs that are branched on are not necessarily on SPT for BBSS. Therefore, arcs inside SPT can change drastically from parent problem to sub-problems, as arcs outside of the SPT are set to included as branching continues. This result in dramatic changes of SPT arcs, and re-optimization proves to be slower than solving from scratch based on our preliminary computational results. Therefore, Dijkstra's algorithm [59] is used to calculate the lower bound from scratch for each sub-problem generated throughout BBSS.

8.5. Stable Set Based Algorithm

Stable set found by the greedy algorithm at iteration t of BBSS is defined as $\mathscr{S}^{(t)}$. Due to our notation, the set of arcs $\mathscr{S}^{(t)} \cup I^{(t)}$ is a feasible solution if its size is $|V(N)| - 1$. The set of arcs $\mathscr{S}^{(t)} \cup I^{(t)}$ is also a stable set of $\widehat{C}$, and it is the set that is important in our implementation. Therefore, the set $\mathscr{S}^{(t)} \cup I^{(t)}$ is represented with $\mathscr{S}$ for the sake of simplicity in notation of pseudo-code of BBSS given as Figure 8.2.

Sorting the active node list in non-decreasing heap of relaxation cost uses a large amount of memory for this BB. Initial lower bounds are too small to prune any tree at the start due to the fact that no information about the cost of an arc is used while branching except for the rare case of optimality branching. Huge size of possible stable

sets is another potential cause for the memory problems. In order to clean the active node list as fast as possible and fix memory problems, active node list heap is sorted in non-increasing relaxation costs. The highest lower bound sub-problem is selected first and it is automatically discarded if it has higher lower bound than the incumbent objective value.

Input: A SPC problem input $N = (V(N), A(N))$, $c_a \geq 0$ for $a \in A(N)$, and an extended conflict graph $\widehat{C} = (V(\widehat{C}), A(\widehat{C}))$ where $V(\widehat{C}) \equiv A(N)$

Output: Best found solution within time limit

begin

Create initial search node 0, $\bar{z} \leftarrow \infty$ ▷ *Initialization*

$\mathcal{L} \leftarrow \{0\}$ ▷ *$\mathcal{L}$ is a non increasing heap of relaxation cost*

while $|\mathcal{L}| > 0$ **and** time limit is **not** reached **do** ▷ *Termination test*

  $\theta \leftarrow$ the first problem in $\mathcal{L}$ ▷ *Problem selection*

  if LB(θ) $< \bar{z}$ **then** ▷ *Cleaning at runtime*

    Find a circuit free stable set $\mathcal{S} \subseteq V(\widehat{C}^{(\theta)})$ using greedy algorithm

    if $|\mathcal{S}| = |V(N)| - 1$ **then** ▷ *Feasible solution*

      Calculate the objective value implied by $\mathcal{S}$ and update $\bar{z}$ if needed

      Create sub-problem $\xi_1, \xi_2, ...$ with optimality branching ▷ *Division*

      for each Sub-problem $\xi \in \{\xi_1, \xi_2, ...\}$ **do**

        if not InfeasTest(ξ) **then** ▷ *Infeasibility test*

          Calculate LB(ξ) using Dijkstra ▷ *Bound calculation*

          $\mathcal{L} \leftarrow \mathcal{L} \bigcup \xi$ ▷ *Add sub-problem to $\mathcal{L}$*

    else ▷ $|\mathcal{S}| < |V(N)| - 1$

      Create sub-problem $\xi_1, \xi_2, ...$ with feasibility branching ▷ *Division*

      for each Sub-problem $\xi \in \{\xi_1, \xi_2, ...\}$ **do**

        if not InfeasTest(ξ) **then** ▷ *Infeasibility test*

          Calculate LB(ξ) using Dijkstra ▷ *Bound calculation*

          $\mathcal{L} \leftarrow \mathcal{L} \bigcup \xi$ ▷ *Add sub-problem to $\mathcal{L}$*

  $\mathcal{L} \leftarrow \mathcal{L} \setminus \theta$ ▷ *Node θ has been processed, remove it from $\mathcal{L}$*

return $\bar{z}$ **and** *corresponding feasible solution* **as the best feasible solution**

end

Figure 8.2. BBSS: Branch-and-bound based on stable sets.

9. COMPUTATIONAL RESULTS

This chapter is devoted to the results of the computational experiments. The settings related to the tests are summarized in Section 9.1. Performance of the benchmark MILP software is discussed in Section 9.2. Finally, results on the performance of the proposed algorithms are reported in Section 9.3 and Section 9.4. The descriptions of the instances and detailed results can be found in Appendices A and B respectively.

9.1. Test Environment

The settings of the computational tests including test conditions, instance generation, performance measures, and implementation details are presented in this section.

9.1.1. Test Conditions

Experiments are performed on a server with Intel Xeon E5-2690-v3 processor running at 2.6 GHz with 64 GB RAM. MILP solver experiments are performed single and multi-threaded, whereas our own algorithms are programmed using C^{++} language and executed in single thread. Weak and strong formulations are solved using Gurobi 9.5.2 solver, which is a well known state-of-the-art commercial software. Moreover, CPU time is limited with $1,800$ seconds for all practical purposes, because optimally solving all instances is very time consuming.

9.1.2. Random Test Instances

Random SPC instances are obtained from randomly generated SP instances according to [64]. First a spanning outtree is created. This is the backbone that ensures the feasibility, both in terms of root connectivity of the network and random conflicts. Then, the rest of the arcs are generated randomly between the vertices, and conflict relations are built between the arc pairs randomly only if both of the arcs are not on

the backbone. Finally, the costs of the arcs, except the ones on the backbone, are selected uniformly from the integer range $[1, 15]$, where 15 is the highest value. The costs of the backbone arcs are set to 15, which gives higher chance to a conflict free shortest path tree other than the backbone; it is selected as an optimal solution in case it is the only feasible solution of the generated SPC instance. A formal outline of this generation scheme is provided in Figure 9.1.

Input: n: vertex number, $|V(N)|$ m: arc number $|A(N)|$, g: conflicting pair number, $|E(C)|$, U: upper bound on arc costs, L: lower bound on arc costs.

Output: a network with at least one feasible solution, its conflict graph, and unit costs, i.e., a feasible SPC instance

begin

1: Set $S \leftarrow \{s\}$, $A \leftarrow \emptyset$,$T \leftarrow \emptyset$
2: **while** $|S| < n$ **do** ▷ *Backbone creation*
3: Select a random vertex $j \notin S$
4: Select a random vertex $i \in S$
5: $A \leftarrow A \cup \{(i, j)\}$, $c_{ij} \leftarrow U$
6: $S \leftarrow S \cup \{j\}$
7: $T \leftarrow A$ ▷ *T is the backbone*
8: **while** $|A| < m$ **do** ▷ *Additional arcs*
9: Select $i \in S, j \in S, i \neq j$ ▷ *Random vertices from S*
10: **if** $(i, j) \notin A$ **then**
11: $A \leftarrow A \cup \{(i, j)\}$
12: $c_{ij} \leftarrow$ random integer $\in [L, U]$
13: $C \leftarrow \emptyset$
14: **while** $|C| < g$ **do** ▷ *Conflict generation*
15: Select $(i, j) \in A, (k, l) \in A, (i, j) \neq (k, l)$ ▷ *Random arcs from A*
16: **if not** $((i, j) \in T$ **and** $(k, l) \in T)$ **then**
17: **if** $\{(i, j), (k, l)\} \notin C$ **and** $\{(k, l), (i, j)\} \notin C$ **then**
18: $C \leftarrow C \cup \{(i, j), (k, l)\}$
19: **return** $A, C, \{c_{ij} : (i, j) \in A\}$

end

Figure 9.1. Instance generation algorithm.

The number of vertices ($|V(N)|$), the percentage of possible arcs (i.e., network density (Γ)), and the percentage of possible conflicts (i.e., conflict density (Ω)) are the parameters of this generation scheme. Given the first two it is possible to determine the number of arcs as $|A(N)| = \left\lfloor \frac{\Gamma(|V(N)|-1)|V(N)|}{100} \right\rfloor$, which is equal to the number of vertices $|V(C)|$ of the conflict graph. Then, the total number of conflicts follows from $|E(C)| = \left\lfloor \frac{\Omega(|A(N)|-1)|A(N)|}{200} \right\rfloor$. Their selected values are $|V(N)| \in \{30, 40, 50, 60, 70\}$, $\Gamma \in \{20, 30, 40, 50, 60, 70, 80\}$ and $\Omega \in \{10, 20, 30, 40\}$. Three instances are generated for each setting with different random seeds and the reported summary results are based on the averages of these three runs. We have realized extensive preliminary computational tests with the new BB algorithm and Gurobi for determining the experimental values of parameters $|V(N)|$, Γ and Ω. They are the most meaningful ones for benchmarking. The instances with $|V(N)| < 30$ and / or $\Gamma < 20$ and / or $\Omega < 10$ and / or $\Omega > 40$ are too easy, and solved by all exact solution methods in a couple of seconds. In the contrary, it is not possible to compute even a feasible solution for $|V(N)| > 70$ and / or $\Gamma > 80$ within $1,800$ secs. CPU time limit. However, the selected values provide a meaningful test problem set helping to demonstrate the differences between the efficiencies of the solution methods clearly.

9.1.3. Realistic Test Instances

Real world SP instances that exist in shared databases are large and the addition of conflict relations to them produces SPC instances unsolvable within reasonable time. This fact oriented us towards small *Asymmetric Traveling Salesman Problem* (ATSP) instances available in TSPLIB [65]. These are "ftv33", "ftv44", "ftv55", "ftv64", and "ftv70". Conflicts are added to them again using an algorithm similar to Figure 9.1 of random instance generation.

The first vertex is selected as the source. An outtree is generated randomly from the source vertex. This tree is treated as backbone when random conflict relations are added to the problem; for example a random conflict relation can be created if and only if they are not present on the backbone together. ATSP networks represented by the

TSPLIB data are complete, which makes generated SPC instances again unsolvable. Hence, after choosing an arc density we have reduced their density by randomly deleting arcs with exception for the ones on the backbone.

Arc densities are selected from $\Gamma \in \{20, 30, 40, 50, 60, 70, 80\}$ and conflict densities are obtained from $\Omega \in \{10, 20, 30, 40\}$. Each setting is replicated three times with different random seeds, which makes 420 instances at grand sum. Summary statistics are based on the averages of three instances with random seeds as before. As a result, the main difference of realistic instances from the random set is the real world cost data on the network.

9.1.4. Performance Measures

We choose running time, i.e., CPU time, as the main measure for assessing the efficiency of the novel algorithms. We define it as the amount of time that passes until the algorithm proves that the incumbent is optimal. Since we do not allow it to be more than $1,800$ secs. in our experiments, we prefer not to consider the ones exceeding this limit. As a consequence, missing values may harm the reliability of the final statistics such as the averages of the CPU times and their variances. Hence, we think it is appropriate to support the comparisons with additional metrics that measure the amount of work accomplished within a fixed amount of time, rather than the time elapsed until the completion of the task. We introduce the number of instances whose proven optimum solution is output within the CPU time limit ("Opt. Sols."), the number of instances for which a feasible solution is output but its optimality is not guaranteed ("Feas. Sols."), the average number of BB tree nodes explored until the algorithms stop ("Exp. Nodes"), and the number of instances for which best known lower bound is reported ("Best LB"). Besides, we calculate two types of averages: "CPU*" incorporates the CPU times of the instances for which optimal solutions are obtained by all of the compared solution methods in common. "CPU" is taken over the complete test set and includes also $1,800$ secs. spent for the cases where a proven optimal solution is not output within the CPU time limit.

Another tool we benefit for benchmarking the efficiency of an algorithm is the *performance profile* of its running time [66]. It is a plot of the cumulative probability of having the running time of an algorithm less than or equal to τ-fold of the running time of the best performing one. Technically speaking, τ is an upper bound on the performance ratio as noted by [66]. The horizontal axis consists of different τ values. The vertical axis represents the fraction of the instances, for which the considered method requires at most τ times larger running times than the best performing one. When we look at the values different performance profiles have on the vertical axis for τ equals to 1, we obtain the upper bounds on the number of wins against the most efficient one of the compared methods. The performance of an algorithm becomes higher as the profile approaches the upper left corner of the figure. We have to point out that the profiles we give are $\log 2$ scaled, i.e., τ values of the horizontal axis should be read as 2^{τ}, which makes the vertical axis values comparable for $\tau = 0$ since $2^0 = 1$.

9.1.5. Implementation Details

Several parameters are experimented on when MILP formulations are implemented using Gurobi 9.5.2. For example, removing "Presolve" helps to improve performance on small instances while making large instances unsolvable. Therefore it is set to its default setting which is "on". Another parameter is "Feasibility Tolerance", reporting wrong solutions in several instances with its default setting of 10^{-6}. This issue is fixed as it is set to 10^{-7}. Lastly, "Thread Count" is set to 4 for multi-threaded runs. Rest of the parameters for Gurobi 9.5.2 are left at their default values.

LEMON, an open source C++ graph template library [67], constitutes an important component of our algorithms. Its "Static Digraph" structure is used in order to represent networks. "Smart Graph" structure is utilized for conflict graphs. "Filter Arcs" and "Filter Nodes" are used excessively in order to represent sub-networks and sub-graphs for sub-problems. LEMON's heap implementation for Dijkstra's algorithm is used whenever a SP solution has to be calculated from scratch.

When a single arc to be removed from a SPT and any SP algorithm (e.g., Dijkstra [59]) is solved from scratch, not many arcs change from the original tree. For example, if the removed arc is an inarc of a leaf vertex of the tree, only a new arc to cover that leaf node is needed to get a new optimal SP tree. In broader terms, depending on the location of the deleted arc on the tree, only the shortest paths on which deleted arc exists changes in the new SPT. Therefore, it might be beneficial to use as much information as possible from the old tree as arcs are removed from SPT. In order to facilitate the primal-dual algorithm, a special tree data structure is used. In its bi-directional tree data structure, necessary tree update functions such as adding and removing arcs, searching an arc or a vertex, or calculating cost from scratch are implemented. Objective cost in primal-dual algorithm can be updated as T_1 grows to span N as minimum reduced cost is incurred for each vertex added to T_1, without the need for objective function calculation from scratch.

In order to assess the effectiveness for primal-dual reoptimization algorithm, data has been collected on 420 realistic instances. On average around 7% of tree arcs are deleted at each call of reoptimization algorithm and this ratio does not depend on the number of vertices of the network. Moreover, the ratio of deleted arcs over the total number of arcs in the network is between 0.5% on small networks and getting as low as 0.2% on large networks, potentially increasing the efficiency of the proposed reoptimization scheme versus solving from scratch as the network size increases.

When multiple arcs are removed from a SPT, primal-dual algorithm is run for each arc in the order of their removal from the tree. This ordering may have an effect on the performance of the restoration algorithm. However, it is not studied in the context of this thesis.

Instead of solving relaxed SP sub-problem for calculating a lower bound, an alternative fast lower bound algorithm is tested. To that end, the sum of costs for included arcs constitute a fast lower bound calculation strategy, compared to solving or re-optimizing SP to optimality. Recall that, any arc can be present in multiple

shortest paths on a SP solution. Preliminary experiments show that lower bounds obtained with the fast lower bound strategy are very weak compared to actual SP objective values, and offer very little value.

There are several alternatives regarding the ordering of arcs O, which is used in the greedy algorithm given as Figure 8.1. For example, arcs can be ordered with respect to their degree sums of their neighbors. Another ordering is to use the degree as primary ordering and the neighbor degrees for tie-breaking. The preliminary results for ordering of O did not show any improvement for neighbor degree strategy. Therefore, non-decreasing degree order is implemented as the default strategy.

Circuit prevention is a key element in both extended diving (Figure 7.2) and in stable set search for BBSS (Figure 8.1). It maintains a disjoint set structure corresponding to included set I. This data structure makes backtracking possible, maintaining a stack of latest additions to the forest, recording the last arc added to the included set. Its aim is to reverse three possible changes discussed in Section 6.1, caused by the addition of a free arc to the included set.

9.2. Performance of the Formulations

As can be remembered we have introduced three MILP formulations of SPC in Chapter 4, which we call strong, weak and clique formulations. They differ in their conflict constraints and use equivalent inequalities previously suggested for stable set formulations. We report the results of our computational experiments in order to select the most efficient one, which is the reference method while benchmarking the performance of the novel algorithms. We do not consider clique formulation for all practical purposes because it can have an exponential number of conflict constraints each of which is related to a maximal clique of the conflict graph. However, both strong and weak formulations use a polynomial number of inequalities of the form given as constraints (4.11) and (4.14) respectively.

A 168-instance subset of the test problems is used to compare the performances of the formulations in order to decide on the reference method for benchmarking. They all have 30 and 40 vertices. Single and multi-threaded options are considered for both of them. The summary of the results can be found in Table 9.1. Columns represent single and multi-threaded runs of weak and strong formulations. Rows include the values of the test statistics. Notice that the running time averages given in the last row, i.e., "CPU*" values, are obtained on the 90 instances whose proven optimal solutions are computed within $1,800$ secs. CPU time limit, considering all of the formulations tested.

Table 9.1. Performance of the formulations.

Test Statistic	**Weak-multi th.**	**Strong-multi th.**	**Weak-single th.**	**Strong-single th.**
Opt. Sols.	109	109	99	91
Feas. Sols.	138	143	127	124
Explored Nodes	59,067	27,088	14,649	7,974
Best LB	133	136	92	85
CPU (secs.)	735	753	863	961
CPU*(secs.)	27	56	126	235

Multi-threaded runs outperform single-threaded counterparts for every performance measure, as expected. They give higher number of best feasible and proven optimal solutions, and tighter lower bounds. On the single-threaded setting, weak formulation outperforms strong formulation with respect to all performance measures. When run on multi-threaded setting, both formulations report the same number of proven optimal solutions. Strong formulation has a slight edge on both the number of feasible solution outputs and best lower bounds. However, weak formulation can be solved more efficiently when compared to strong formulation as can be seen from the last two rows of Table 9.1. A similar observation can also be made from *performance profile* depicted with Figure 9.2. The winning rate of the formulations can be read for $\tau = 0$. It is 0.68 for the multi-threaded versions, which becomes 0.45 and 0.35 respectively for weak and strong formulations' single-threaded runs. Besides, the cumulative probability of Weak-multi th. stabilizes approximately at 0.99 for $\tau = 2.4$, which is

slightly higher than Strong-multi th.'s, and considerably higher than Weak-single th.'s and Strong-single th.'s; these are respectively 0.98, 0.78 and 0.70.

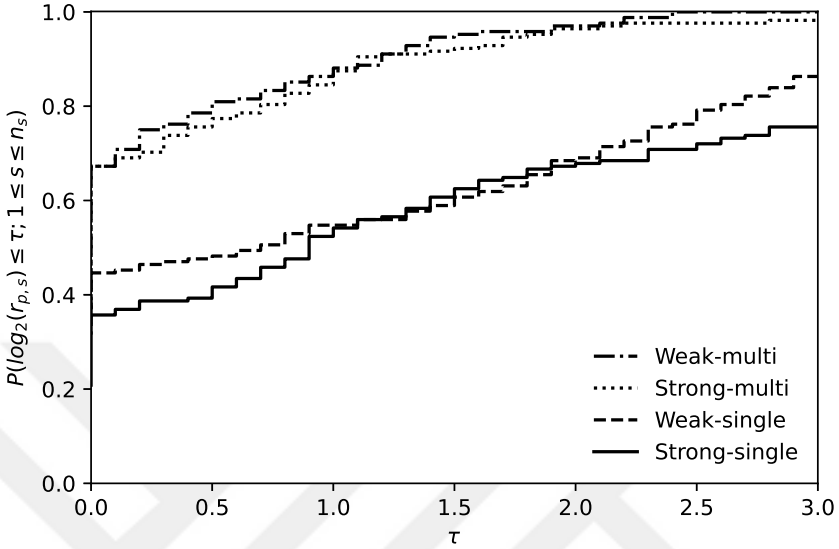


Figure 9.2. Performance of the MILP formulations.

Although it gives the weakest (i.e., smallest) LP relaxation lower bound, weak formulation explores considerably larger number of nodes of the search tree, because its LP relaxation is solved faster. This provides a chance to discover many more feasible solutions, and decreases the upper bound and tightens the lower bound. However, the situation is opposite for the strong formulation, which gives a smaller chance to detect new feasible solutions with a smaller potential to improve the upper bound and tighten the lower bound faster. This suggests that the efficiency of a lower bound method is more important than its accuracy. Hence, we choose the results obtained by solving weak formulation with Gurobi 9.5.2 in the multi-threaded setting as benchmarks, to evaluate the performance of the novel algorithms.

9.3. Performance of the Novel Algorithms in Random Instances

BB based on conflicts is tested only with the conflicting arc branching rule, since it has the highest potential of generating the smallest search tree, as follows from the brief analysis we have made in Section 6.2.4. Similarly, infeasibility test has a potential to reduce the total number of explored nodes, as explained in Section 6.6.

Then, BB based on conflicts using arc branching with (*BBA*) and without (*BBA w/o Inf.*) the infeasibility test, the diving algorithm (*Diving*) and the stable set based BB with infeasibility test (*BBSS*), all running on single thread, is compared with the best version of the MILP formulation (*Weak - multi*), and (*Weak - single*), running on 4 and 1 threads respectively. Results are reported with Table 9.2.

Table 9.2. Performance of the new algorithms: random instances.

Test Statistic	**BBA**	**BBA w/o Inf.**	**Diving**	**Weak - single**	**Weak - multi**	**BBSS**
Opt. Sols.	193	146	186	127	159	144
Feas. Sols.	194	147	203	180	217	209
Exp. Nodes	1,820,343	3,248,743	397,200,645	8,575	40,176	2,353,489
Best LB	344	238	0	127	160	144
CPU (secs.)	1,044	1,232	1,067	1,335	1,183	1,240
CPU* (secs.)	4.9	83.1	17.0	240.5	49.9	29.9

Conflicting arc branching when used in conjunction with the infeasibility test, outperforms the MILP formulation across the board. Among 420 randomly generated test instances, BBA is able to find an optimal solution in 193, compared to 159 of Weak-multi th. The average number of nodes it explores is 45 times larger than that of multi-threaded MILP formulation, and the lower bound it computes is the tightest among all algorithms for 344 instances. Only in three out of 420 instances, weak formulation is able to provide a better lower bound than BBA. It is faster than "Weak - multi" with $1,044$ vs. $1,183$ and 4.9 vs. 49.9 seconds averages, respectively for "CPU" and "CPU*". Recall that CPU averages are taken over the complete test set and includes $1,800$ seconds time limit for instances whose solution's optimality is unproven. CPU* values are taken over 98 instances whose proven optimal solutions are computed within the time limit for each one of six algorithm settings. BBA and Weak - multi both obtain optimum solutions in common 158 instances and CPU averages for those instances are 40 and 163 respectively. Recall that, Weak - multi is able to find 159 optimum solutions at sum and in 158 of those solutions, BBA is also able to find the optimum solution, and doing so four times faster on the average.

Second best contender for the number of optimum solutions obtained is Diving with 186, which is also higher than that of MILP formulation of 159. As expected the average number of explored nodes is very high compared to others, 1000 times higher than Weak - multi. Unfortunately we do not obtain any lower bound with Diving. However, it is more efficient than best MILP formulation with $1,067$ vs. $1,183$ and 17 vs. 49.9 seconds averages, respectively for "CPU" and "CPU*". Diving and Weak - multi both obtain optimum solutions in common 137 instances and CPU averages for those instances are 43 and 170 respectively independent of other algorithms.

BBSS's performance is lackluster compared to both BBA and Diving. Still, its performance is comparable with the best MILP formulation, Weak - multi. On total number of optimum solutions found, BBSS is less than that of Weak - multi, 144 vs. 159. We see similar story on the number of feasible solutions found 208 vs. 217, best lower bound found 143 vs. 160, and "CPU" averages $1,250$ vs $1,183$. However, BBSS is more than twice as fast on the average of 98 commonly solved instances, since "CPU*" averages are 22.8 and 49.9 respectively. BBSS is exploring around the same number of nodes as BBA, which is around 55 times of MILP formulation's. BBSS and Weak - multi both obtain optimum solutions in common for 140 instances and CPU averages for those instances are 168 and 155 respectively, confirming comparable performance of these two, independent of other algorithms. Compared to BBA, BBSS performs worse. BBA is able to find the optimum solution on all of the 144 instances for which BBSS found an optimal solution. The CPU averages for those instances are 33 vs. 165 where BBA is five times faster on the average.

BBA w/o Inf. formulation performs the worst among the new algorithms showing the importance of the infeasibility test on the performance of conflict based BB. Averages also point to the improvement introduced by the infeasibility test on BBA: "CPU" reduces from $1,232$ to $1,044$ and "CPU*" reduces from 83.1 to 4.9. Despite performing the worst across the board for all metrics, it provided the second best number of "Best LB" signifying the importance of SLB sub-problem selection rule. Figure 9.3 shows the clear dominance of BBA with the infeasibility test over its plain version. Therefore,

BBA without the infeasibility test is not considered as an alternative and disregarded from the analysis of the *performance profile* in Figure 9.4 and not experimented on realistic instances in Section 9.4.

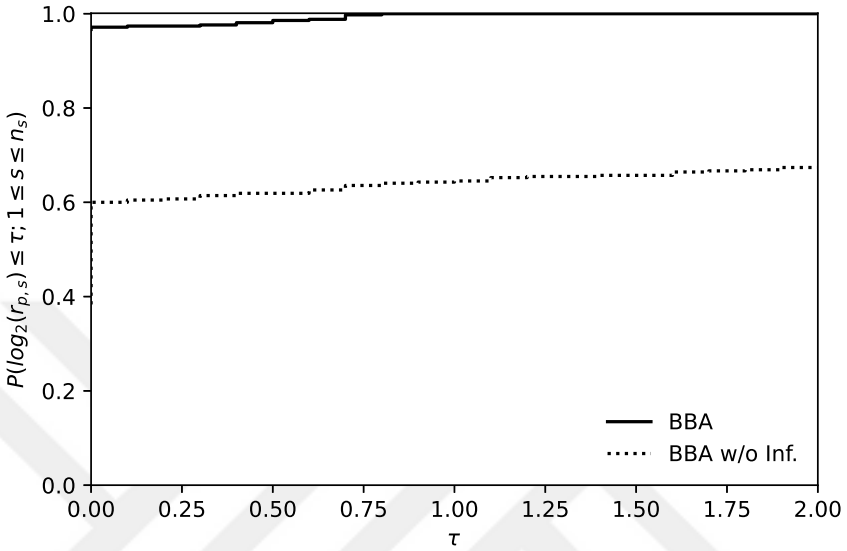


Figure 9.3. The effect of the infeasibility test on BBA.

Number of feasible solutions is the only statistic "Weak - multi" is better. However we need to keep in mind that it is the only multi-threaded algorithm. Every one of proposed single-threaded algorithms is better in every test statistic compared to single-threaded MILP formulation "Weak - single".

Looking at *performance profiles*, in Figure 9.4, a similar observation can be made about the performance of the new algorithms. The winning rate of the methods can be read for $\tau = 0$. It is 0.83 for Diving, which becomes 0.71 for BBA and around 0.53 for others. Besides, the cumulative probability of BBA stabilizes approximately at 0.99, which is followed by Diving and BBSS with 0.9 and 0.86 respectively for $\tau = 5.5$. Hence, the novel algorithms outperform the best MILP formulation both on winning rate and the cumulative probability of winning. At sum, considering the results incorporated with Table 9.2 and *performance profiles* illustrated with Figure 9.4, and keeping in mind that proposed algorithms are single-threaded, we can say that our two best single-threaded algorithms outperform multi-threaded Gurobi 9.5.2 solution of the weak formulation and the third best algorithm BBSS is comparable with it at worst.

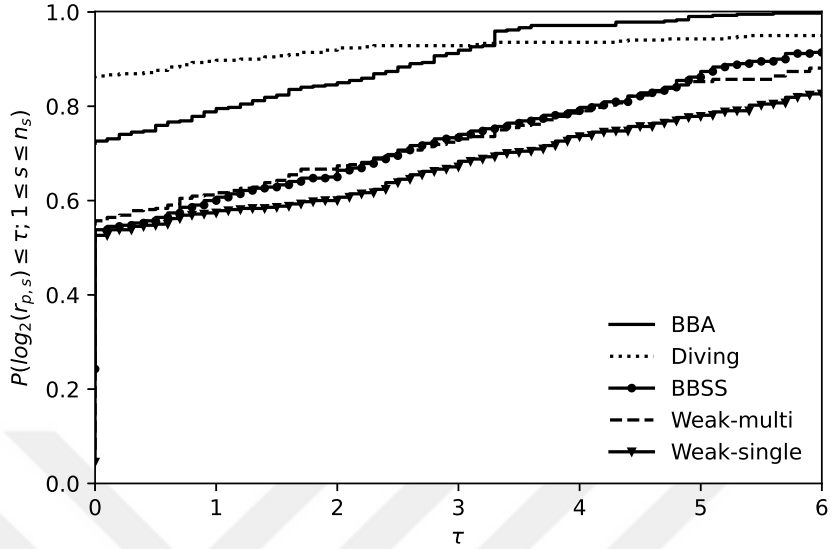


Figure 9.4. Performance of the algorithms on random test instances.

If we pairwise compare BBA and Diving, we see that they both are able to find the optimum solution on common 161 instances and CPU averages for those instances are 117 and 63 respectively. Looking from that perspective, Diving may seem to be faster than BBA. Initial pairwise performance of Diving compared to BBA can also be seen on the *performance profile* in Figure 9.4, starting ahead with respect to the winning probability at $\tau = 0$. However, there are 32 instances where BBA finds optimum solutions and Diving does not and in 25 instances Diving finds optimum solutions and BBA does not. It is those observations that make the difference so that BBA is winning in average solution times overall shown in the "CPU" row in Table 9.2. This observation is also represented on the performance profile in Figure 9.4, where BBA surpasses Diving cumulative winning probability at $\tau \approx 3.2$.

9.4. Performance of the Novel Algorithms in Realistic Instances

Second part of experimentation continues here on realistic instances, obtained from TSPLIB [65] as explained in Section 9.1.3. Best version of our BBA is the one with the infeasibility test as shown with the tests done on random instances. Similarly, the best version of MILP is Weak - multi. Therefore, experiments with BBA w/o

Inf. and Weak - single are not repeated on realistic test instances due to the fact that they are dominated by their better versions. Experiments on the realistic instances are summarized with Table 9.3.

Table 9.3. Performance of the new algorithms: realistic instances.

Test Statistic	**BBA**	**Diving**	**Weak - multi**	**BBSS**
Opt. Sols.	207	189	130	112
Feas. Sols.	207	205	193	133
Explored Nodes	1,751,912	363,601,564	46,693	720,044
Best LB	411	0	132	113
CPU (secs.)	1,016	1,057	1,301	1,374
CPU* (secs.)	4.2	17.3	104.5	156.4

BBA outperforms other algorithms on the realistic instances according to the performance measures shown in Table 9.3. Among 420 realistic test instances, BBA is able to find an optimal solution in 207, where second best algorithm is Diving with 189 compared to Weak - multi's 130. BBA is the best on the number of feasible solutions with 207 followed by Diving with 205 and Weak - multi with 193. The average number of nodes BBA explores is 37 times larger compared to MILP formulation, and the lower bound it computes is the tightest for 411 instances out of 420. BBA is more efficient with $1,016$ vs. $1,057$ and 4.2 vs. 17.3 seconds averages, respectively for "CPU" and "CPU*" compared to second best algorithm Diving. Recall that CPU averages are taken over the complete test set and includes $1,800$ secs. time limit for instances whose solution's optimality is unproven. "CPU*" values are taken over 105 instances whose proven optimal solutions are computed within the time limit by all four algorithms tested. The advantage of MILP formulation is lost on the realistic instances: BBA is better than Weak - multi also on the number of feasible solutions found.

A similar observation can also be made from the *performance profiles* provided in Figure 9.5. The winning rate of the methods can be read for $\tau = 0$. It is over 0.84 for Diving, which becomes 0.68 for BBA. The cumulative probability both algorithms

stabilize approximately at 0.99 for $\tau = 4.5$, which is considerably higher the other two algorithms which are around 0.68. At sum, considering the results incorporated with Table 9.3 and *performance profiles* illustrated with Figure 9.5, we can easily say out of three proposed algorithms, which are implemented as single-threaded, at least two are outperforming multi-threaded MILP of the weak formulation on realistic instances.

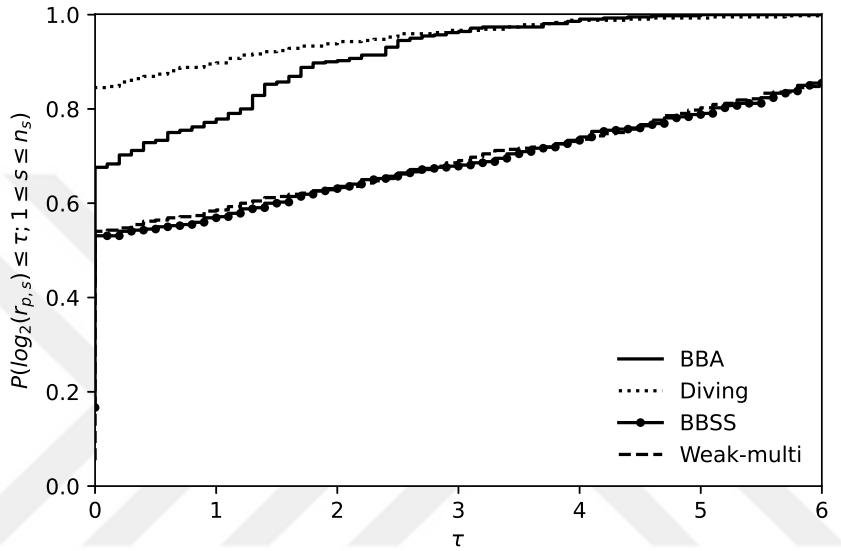


Figure 9.5. Performance of the algorithms on realistic test instances.

BBA and Diving both solve all 112 instances BBSS solved to optimality. CPU averages on those instances are 5.1, 18, and 204 secs. for BBA, Diving, and BBSS respectively. On 25 instances Weak - multi finds optimal solution where BBSS does not, whereas on 7 instances BBSS gives an optimal solution in given time where Weak - multi does not. Hence, relative comparable performance of the BBSS to Weak - multi is lost on realistic instances. A possible explanation for this could be real world cost information on realistic instances, (realistic arc costs are higher compared to the ones of random instances) and BBSS is the most sensitive to higher arc costs compared to other algorithms.

In common 115 instances, where Diving and Weak - multi both solve to optimality, CPU averages are 37 and 150 seconds respectively. BBA and Diving solve to optimality in common 184 instances where CPU averages are 123 and 129 seconds respectively.

Performance of the algorithms on realistic test instances can be summarized as follows: BBA performs the best in the number of optimum and feasible solutions found and faster in terms of average solution times. Diving is the second best performer across the same performance metrics. Weak - multi, which is the only multi-threaded algorithm, performs second to last, performing marginally better than BBSS.

9.5. The Effect of Graph and Conflict Densities

Test instance parameters play an important role on the difficulty of a random instance. In order to analyze their effect, six tables are prepared. Table 9.4 and Table 9.5 includes the effect of the number of network vertices, Table 9.6 and Table 9.7 are for the effect of network density, and Table 9.8 and Table 9.9 expose the effect of conflict density on both randomly generated and realistic instances separately.

Table 9.4. The number of optimally solved instances for various $|V(N)|$ values: random instances.

V(N)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
30	80	54	70	55
40	44	43	39	39
50	31	37	23	23
60	23	28	17	18
70	15	24	10	9
Total	193	186	159	144

Difficulty of an instance increases as the number of arcs increases due to increasing vertex count or network density, which can be seen in Table 9.4 and Table 9.6 for random instances, and in Table 9.5 and Table 9.7 for realistic instances.

Conflict density however, affects the difficulty of an instance in a different way compared to vertex number and network density. At first glance, the difficulty seems to decrease as conflict density is increased. However, looking at the differences between conflict densities 10 and 20 we do not actually see the similar increase in the number of optimum solutions found between 20 and 30. This makes sense as a SP, without the

conflict relations, is polynomially solvable [59]. If SP is at its easiest setting without the conflict relations and we show empirically the problem becomes easier as the number of conflict relations increase, there should be a point in which increasing the number of conflict relations for a given instance stops making the instance harder to solve and it becomes easier. In fact, we have empirically observed that, for a given vertex number and network density, an instance reaches its top difficulty (i.e., running time becomes the longest possible) for $\Gamma \approx 2.5\Omega$ approximately, looking at the average solution times. In fact, as mentioned in [68], it is experimentally observed that the hardest instances of MS and MWS are the ones with low density. As a consequence, one could expect a similar relation between the conflict density and solution difficulty of SPC.

Table 9.5. The number of optimally solved instances for various $|V(N)|$ values: realistic instances.

V(N)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
33	70	55	60	48
44	46	42	28	28
55	39	35	19	18
64	29	30	13	9
70	23	27	10	9
Total	207	189	130	112

Table 9.6. The number of optimally solved instances for various graph density (Γ) values: random instances.

Γ(%)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
20	50	47	48	48
30	40	37	37	36
40	32	30	26	23
50	24	24	17	16
60	18	21	13	9
70	16	15	12	9
80	13	12	6	3
Total	193	186	159	144

Table 9.7. The number of optimally solved instances for various graph density (Γ) values: realistic instances.

Γ(%)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
20	50	48	47	42
30	42	36	33	28
40	36	33	19	18
50	29	24	13	9
60	20	21	6	9
70	16	15	7	3
80	14	12	5	3
Total	207	189	130	112

Table 9.8. The number of optimally solved instances for various conflict density (Ω) values: random instances.

Ω(%)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
10	25	3	24	9
20	33	24	24	22
30	55	64	48	47
40	80	95	63	66
Total	193	186	159	144

Table 9.9. The number of optimally solved instances for various conflict density (Ω) values: realistic instances.

Ω(%)	**BBA**	**Diving**	**Weak - 4**	**BBSS**
10	18	3	18	3
20	32	25	22	16
30	64	65	36	36
40	93	96	54	57
Total	207	189	130	112

Diving and BBSS are the worst performers in low conflict densities compared to other algorithms because of the increased number of feasible stable sets. These algorithms are based on stable sets, hence increasing the number of possible stable sets affects them the most.

Looking at the similar behavior of our algorithms both on random and realistic instances, we can say that our algorithms' performance does not depend on the type of the instance except for BBSS. Algorithms perform better as conflict density is increased in general. However, Diving favors higher Ω values more than others as seen in Table 9.8 and Table 9.9. That is to say, one can propose to use BBA on low conflict density instances $(10, 20)$ and switch to using Diving when solving an instance with high conflict densities $(30, 40)$.

10. CONCLUSION

We have studied a recent extension of the one-to-many shortest path problem which includes additional disjunctive conflict constraints (SPC). It involves in finding a shortest path tree whose arcs correspond to a stable set of the conflict graph, which is known to be $\mathcal{NP}$-hard in contrast to polynomially solvable SP. As a result, the feasible solution set of the MILP formulation of SPC consists of conflict free spanning outtrees, which are represented by the flow balance equalities (i.e., path equalities) on the vertices of the original network, and the packing inequalities on the vertices of the conflict graph (i.e., arcs of the original network) in addition to the integrality restrictions.

After basic concepts and notation are given in Chapter 2, previous related work such as paths avoiding forbidden pairs, paths avoiding forbidden transitions, and properly edge colored paths are summarized in Chapter 3. The existence of different representations of the conflict constraints in the stable set literature makes different formulations of the problem possible. They are explained in detail in Chapter 4. Three formulations with different strengths according to their LP relaxation bounds are tested. As a consequence of the computational experiments realized with a large set of test instances, we can say surprisingly that the weakest of the MILP formulations is the most efficient when solved by Gurobi 9.5.2's MILP solver. Because of the lesser number of constraints in the weak formulation, its LP relaxation solution times is faster than that of the strong formulation. This in turn allows the weak formulation to calculate more BB search nodes resulting in better performance.

In order to generate alternative solution methods to MILP, one possible method is to solve SP as a relaxed problem and then to enforce conflict restrictions by means of division as a BB algorithm. Then, multiple SP problems need to be solved for each branch generated this way. Therefore, a reoptimization technique is developed and its

reasoning is explained in Chapter 5. In principle, it is a primal-dual method that fixes primal and dual infeasibility created as a result of the division at a search node.

One of the new algorithms is BBA, which is explained in detail in Chapter 6. This BB essentially benefits from the information provided by the SP relaxations during its search through the solution space. This consists of the lower bound and a conflicting arc pair that exists in the optimal relaxed solution, i.e., SPT. Three branching strategies are considered. They are all dichotomized and based on the prohibition of the selected conflicting arc pair in the child problems. As the result of a brief analysis, conflicting arc branching seems to be the most efficient and it is selected for implementation. We have exploited particular features of SP and implemented it carefully in support with special data structures to increase the efficiency utilizing the primal-dual re-optimization algorithm developed in Chapter 5.

Second new algorithm is a depth first search Diving explained in Chapter 7. Diving focuses on finding a feasible solution as fast as possible, iterating over all possible solutions. Although it is very efficient with respect to solution times, memory consumption is exhaustive compared to other algorithms. In order to alleviate its memory problems, an extended version is developed and tested. However, complicating the algorithm with additional data structures to implement backtracking mechanisms or any other improvement mechanism on diving proves that it is not worth. The simplest depth first search for diving performs better compared to the extended version.

Third new algorithm BBSS, which focuses on the stable sets of the conflict graph, is developed in Chapter 8. Search strategy of BBSS is the opposite of BBA's, which finds spanning outtree first and deals with conflicts later. Instead, a stable set of the conflict graph is found at each sub-problem, naturally satisfying conflict restrictions. However, that stable set is neither of full size nor correspond to a valid SPT necessarily. Therefore, BBSS requires different branching rules for each case. BBSS is further improved by using Dijkstra's algorithm for generating lower bounds, clique extension of stable sets for calculating W, and reduced cost branching. With extra improve-

ments, BBSS becomes a comparable algorithm to the baseline multi-threaded MILP formulation.

We have assessed the performance of new algorithms by benchmarking their efficiency with best MILP formulation using with Gurobi 9.5.2's MILP solver. Single-threaded implementations of BBA and Diving are both faster than even the multi-threaded MILP formulation (Weak - multi) based on extensive computational analysis on a total of 840 randomly generated and realistic instances. The attempt for narrowing the solution space by the infeasibility test has been paid back in both BB formulations; its impact on the running time has been particularly remarkable. As a clear best performer in both random and realistic instances, BBA is both a memory and processing power efficient algorithm running on single thread, outperforming the best MILP formulation solved by Gurobi 9.5.2 solver using multiple threads.

Diving, as an alternative solution methodology, performs well on instances with high conflict ratios compared to other algorithms. However, as number of conflicts are reduced, it is the most unfavorably affected. If memory consumption is not a problem, one could propose to use Diving as a preferred solution method for instances with high conflict density ($\Omega \geq 30$). One could say, memory is not a problem anymore due to cloud computing developments in the last decade. However, unless *zero-one* memory language evolves into continuous representation with the advent of reliable *Quantum Computers*, memory limitations for SPC or any other combinatorial problem, will continue to exist as problem sizes grow with current computer technology.

BB based on stable sets of the conflict graph, BBSS, performed the worst among the new algorithms. However, it is still comparable to the best MILP formulation. Although it is not the best performer in its current form, we find the discussion and methodology beneficial in terms of understanding the nature of SPC itself. The selection of $\mathscr{S}$ is consequential for the performance of the algorithm. There is a relation between the selection of the stable set $\mathscr{S}$ and the selection of the maximal sub-graph W while feasibility branching. While a maximum cardinality $\mathscr{S}$ is important for obtain-

ing feasible solutions, selection of $\mathscr{S}$ maximizing the size of W increases the efficiency of the algorithm by reducing the number of branches, where cardinality of $\mathscr{S}$ is not $|V(N)| - 1$.

Performance of the novel algorithms can be further improved by a multi-threaded implementation as a future research direction. BBSS algorithm may be improved further by alternative relaxation, e.g. LP relaxation, or branching ideas. LP relaxation can be used to develop branch-and-cut algorithms for SPC. Based on the primal-dual algorithm, a generalized dynamic SPT reoptimization algorithm can be another future research direction. Beam search and truncated BB are also potential future research ideas for efficient solution of SPC.

All in all, SPC is studied extensively in this thesis, providing relative literature and formulations. Efficient solution methodologies are developed and extensively tested. To the best of our knowledge, the best purpose built algorithm for SPC is BBA.

REFERENCES

1. Sun, M., "A Tabu Search Heuristic Procedure for Solving the Transportation Problem with Exclusionary Side Constraints", *Journal of Heuristics*, Vol. 3, No. 4, pp. 305–326, 1998.

2. Sun, M., "The Transportation Problem with Exclusionary Side Constraints and Two Branch-and-Bound Algorithms", *European Journal of Operational Research*, Vol. 140, No. 3, pp. 629–647, 2002.

3. Goossens, D. R. and F. C. R. Spieksma, "Structured Modeling: Survey and Future Research Directions", *4OR*, Vol. 7, No. 1, pp. 51–60, 2009.

4. Darmann, A., U. Pferschy and J. Schauer, "Determining a Minimum Spanning Tree with Disjunctive Constraints", *Algorithmic Decision Theory: First International Conference*, Vol. 5783, pp. 414–423, Venice, Italy, 2009.

5. Zhang, R., S. N. Kabadi and A. P. Punnen, "The Minimum Spanning Tree Problem with Conflict Constraints and Its Variations", *Discrete Optimization*, Vol. 8, No. 2, pp. 191–205, 2011.

6. Samer, P. and S. Urrutia, "A Branch and Cut Algorithm for Minimum Spanning Trees under Conflict Constraints", *Optimization Letters*, Vol. 9, No. 1, pp. 41–55, 2015.

7. Samer, P. and D. Haugland, "Polyhedral Results and Stronger Lagrangean Bounds for Stable Spanning Trees", *Optimization Letters*, Vol. 17, No. 6, pp. 1317–1335, 2023.

8. Carrabs, F., R. Cerulli, R. Pentangelo and A. Raiconi, "Minimum Spanning Tree with Conflicting Edge Pairs: A Branch-and-Cut Approach", *Annals of Operations Research*, Vol. 298, No. 1, pp. 65–78, 2021.

9. Carrabs, F. and M. Gaudioso, "A Lagrangian Approach for the Minimum Spanning Tree Problem with Conflicting Edge Pairs", *Networks*, Vol. 78, No. 1, pp. 32–45, 2021.

10. Öncan, T., R. Zhang and A. P. Punnen, "The Minimum Cost Perfect Matching Problem with Conflict Pair Constraints", *Computers and Operations Research*, Vol. 40, No. 4, pp. 920–930, 2013.

11. Öncan, T. and İ. K. Altınel, "A Branch-and-Bound Algorithm for the Minimum Cost Bipartite Perfect Matching Problem with Conflict Pair Constraints", *Electronic Notes in Discrete Mathematics*, Vol. 64, pp. 5–14, 2018.

12. Öncan, T., Z. Şuvak, M. H. Akyüz and İ. K. Altınel, "Assignment Problem with Conflicts", *Computers and Operations Research*, Vol. 111, pp. 214–229, 2019.

13. Akyüz, M. H., İ. K. Altınel and T. Öncan, "Maximum Weight Perfect Matching Problem with Disjunctive Constraints", *Networks*, Vol. 81, No. 4, pp. 465–489, 2023.

14. Pferschy, U. and J. Schauer, "The Maximum Flow Problem with Disjunctive Constraints", *Journal of Combinatorial Optimization*, Vol. 26, No. 1, pp. 109–119, 2013.

15. Şuvak, Z., İ. K. Altınel and N. Aras, "Exact Solution Algorithms for the Maximum Flow Problem with Additional Conflict Constraints", *European Journal of Operational Research*, Vol. 287, No. 2, pp. 410–437, 2020.

16. Altınel, İ. K., N. Aras, Z. Şuvak and Z. C. Taşkın, "Minimum Cost Noncrossing Flow Problem on Layered Networks", *Discrete Applied Mathematics*, Vol. 261, pp. 2–21, 2019.

17. Şuvak, Z., İ. K. Altınel and N. Aras, "Minimum Cost Flow Problem with Conflicts", *Networks*, Vol. 78, No. 4, pp. 421–442, 2023.

18. Ahuja, R. K., T. K. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, New Jersey, 1993.

19. Vielma, J. P. and G. L. Nemhauser, "Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints", *Mathematical Programming*, Vol. 128, No. 1, pp. 49–72, 2011.

20. Darmann, A., U. Pferschy, J. Schauer and G. J. Woeginger, "Paths, Trees and Matchings under Disjunctive Constraints", *Discrete Applied Mathematics*, Vol. 159, No. 16, pp. 1726–1735, 2011.

21. Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Co., New York, 1979.

22. Krause, K., R. Smith and M. Goodwin, "Optimal Software Test Planning Through Automated Network Analysis", *Symposium on Computer Software Reliability*, Vol. 40741-9CSR, pp. 18–22, New York, USA, 1973.

23. Gabow, H., S. Maheshwari and L. Osterweil, "On Two Problems in the Generation of Program Test Paths", *Transactions on Software Engineering*, Vol. SE-2, No. 3, pp. 227–231, 1976.

24. Srimani, P. and B. Sinha, "Impossible Pair Constrained Test Path Generation in a Program", *Information Sciences*, Vol. 28, No. 2, pp. 87–103, 1982.

25. Yinnone, H., "On Paths Avoiding Forbidden Pairs of Vertices in a Graph", *Discrete Applied Mathematics*, Vol. 74, No. 1, pp. 85–92, 1997.

26. Kolman, P. and O. Pangrác, "On the Complexity of Paths Avoiding Forbidden Pairs", *Discrete Applied Mathematics*, Vol. 157, No. 13, pp. 2871–2876, 2009.

27. Chen, T., M.-Y. Kao, M. Tepel, J. Rush and G. Church, "A Dynamic Programming Approach to De Novo Peptide Sequencing via Tandem Mass Spectrometry",

Journal of Computational Biology, Vol. 8, No. 3, pp. 325–337, 2001.

28. Kovac, J., J. Vinar and B. Brejova, "Predicting Gene Structures from Multiple RT-PCR Tests", *9th International Conference on Algorithms in Bioinformatics*, pp. 181–193, Philadelphia, USA, 2009.

29. Kováč, K., "Complexity of the Path Avoiding Forbidden Pairs Problem Revisited", *Discrete Applied Mathematics*, Vol. 161, No. 10, pp. 1506–1512, 2013.

30. Brückner, M., *On the Shortest Path Problem with Pair Constraints*, M.S. Thesis, Freie University, Berlin, 2015.

31. Blanco, M., R. Bonrdörfer, M. Brückner, N. Hoàng and T. Schlechte, "On the Path Avoiding Forbidden Pairs Polytope", *Electronic Notes in Discrete Mathematics*, Vol. 50, pp. 343–348, 2015.

32. Kann, V., "Polynomially Bounded Minimization Problems That Are Hard to Approximate", *Nordic Journal of Computing*, Vol. 1, No. 3, pp. 317–331, 1994.

33. Ferone, D., P. Festa and M. Salani, "Branch and Bound and Dynamic Programming Approaches for the Path Avoiding Forbidden Pairs Problem", *Optimization and Decision Science: ODS, Virtual Conference*, pp. 227–235, Rome, Italy, 2020.

34. Cerulli, R., F. Guerriero, E. Scalzo and C. Sorgente, "Shortest Paths with Exclusive-Disjunction Arc Pairs Conflicts", *Computers and Operations Research*, Vol. 152, p. 106158, 2023.

35. Kotzig, A., "Moves without Forbidden Transitions in a Graph", *Matematický Časopis*, Vol. 18, No. 1, pp. 76–80, 1968.

36. Bernstein, G., Y. Lee, A. Gavler and J. Martensson, "Modeling WDM Wavelength Switching Systems for use in GMPLS and Automated Path Computation", *Journal of Optical Communications and Networking*, Vol. 1, No. 1, pp. 187–195, 2009.

37. Chen, Y., N. Hua, X. Wan, H. Zhang and X. Zheng, "Dynamic Lightpath Provisioning in Optical WDM Mesh Networks with Asymmetric Nodes", *Photonic Network Communications*, Vol. 25, No. 3, pp. 166–177, 2013.

38. Jaumard, B. and D. Kien, "Optimizing ROADM Configuration in WDM Networks", *16th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pp. 1–7, Funchal, Portugal, 2014.

39. Gutierrez, E. and A. Medaglia, "Labeling Algorithm for the Shortest Path Problem with Turn Prohibitions with Application to Large-Scale Road Networks", *Annals of Operations Research*, Vol. 157, No. 1, pp. 169–182, 2008.

40. Bellitto, T., "Separating Codes and Traffic Monitoring", *Theoretical Computer Science*, Vol. 717, pp. 73–85, 2018.

41. Bellitto, T., S. Li, K. Okrasa, M. Pilipczuk and M. Sorge, "Separating Codes and Traffic Monitoring", *Algorithmica*, Vol. 85, No. 5, pp. 1202–1250, 2023.

42. Szeider, S., "Finding Paths in Graphs Avoiding Forbidden Transitions", *Discrete Applied Mathematics*, Vol. 126, No. 2-3, pp. 261–273, 2003.

43. Gourves, L., A. Lyra, C. Martinhon and J. Monnot, "Complexity of Trails, Paths and Circuits in Arc-Colored Digraphs", *Discrete Applied Mathematics*, Vol. 161, No. 6, pp. 819–828, 2013.

44. Kanté, M., F. Moataz, B. Momége and N. Nisse, "Finding Paths in Grids with Forbidden Transitions", *International Workshop on Graph-Theoretic Concepts in Computer Science WG*, Vol. 9224, pp. 154–168, Garching, Germany, 2016.

45. Dorninger, D., "Hamiltonian Circuits Determining the Order of Chromosomes", *Discrete Applied Mathematics*, Vol. 50, No. 2, pp. 159–168, 1994.

46. Abouelaoualim, A., K. Das, L. Faria, Y. Manoussakis, C. Martinhon and R. Saad,

"Paths and Trails in Edge-colored Graphs", *Theoretical Computer Science*, Vol. 409, No. 3, pp. 497–510, 2008.

47. Lyra, A., *On Paths and Trails in Edge-Colored Graphs and Digraphs*, Ph.D. Thesis, Universidade Federal Fluminense, 2009.

48. Amaldi, E., G. Galbiati and F. Maffioli, "On Minimum Reload Cost Paths, Tours, and Flows", *Networks*, Vol. 57, No. 3, pp. 254–260, 2011.

49. Padberg, M. W., "On the Facial Structure of Set Packing Polyhedra", *Mathematical Programming*, Vol. 5, No. 1, pp. 199–215, 1973.

50. Rebennack, S., M. Oswald, D. O. Theis, H. Seitz, G. Reinelt and P. M. Pardalos, "A Branch and Cut Solver for the Maximum Stable Set Problem", *Journal of Combinatorial Optimization*, Vol. 21, No. 4, pp. 434–457, 2011.

51. Della Croce, F. and R. Tadei, "A Multi-KP Modeling for the Maximum Clique Problem", *European Journal of Operational Research*, Vol. 73, No. 3, pp. 555–561, 1994.

52. Letchford, A. N., F. Rossi and S. Smriglio, "The Stable Set Problem: Clique and Nodal Inequalities Revisited", *Computers and Operations Research*, Vol. 123, p. 105024, 2020.

53. Pallottino, S. and M. G. Scutellà, "Dual Algorithms for the Shortest Path Tree Problem", *Networks*, Vol. 29, No. 2, pp. 125–133, 1997.

54. Pallottino, S. and M. G. Scutellà, "A New Algorithm for Reoptimizing Shortest Paths when the Arc Cost Change", *Operations Research Letters*, Vol. 31, No. 2, pp. 149–160, 2003.

55. Nguyen, S., S. Pallottino and M. G. Scutellà, "A New Dual Algorithm for Shortest Path Reoptimization", *Transportation and Network Analysis: Current Trends:*

Miscellanea in honor of Michael Florian, pp. 221–235, 2002.

56. Chan, E. P. and Y. Yang, "Shortest Path Tree Computation in Dynamic Graphs", *Transactions on Computers*, Vol. 58, No. 4, pp. 541–557, 2008.

57. Ferone, D., P. Festa, A. Napoletano and T. Patore, "Reoptimizing Shortest Paths: From State of the Art to New Recent Perspectives", *18th International Conference on Transparent Optical Networks*, pp. 1–5, Trento, Italy, 2016.

58. Festa, P., S. Fugaro and F. Guerriero, "Shortest Path Reoptimization vs Resolution from Scratch: A Computational Comparison", *Optimization Methods and Software*, Vol. 37, No. 3, pp. 1122–1144, 2022.

59. Dijkstra, E., "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

60. Fredman, M. and R. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Journal of the Association for Computing Machinery*, Vol. 34, No. 3, pp. 596–615, 1987.

61. Balas, E. and C. S. Yu, "Finding a Maximum Clique in an Arbitrary Graph", *SIAM Journal on Computing*, Vol. 15, No. 4, pp. 1054–1068, 1986.

62. Balas, E. and J. Xue, "Weighted and Unweighted Maximum Clique Algorithms with Upper Bounds from Fractional Coloring", *Algorithmica*, Vol. 15, No. 5, pp. 397–412, 1996.

63. Balas, E. and J. Xue, "Minimum Weighted Coloring of Triangulated Graphs, with Application to Maximum Weight Vertex Packing and Clique Finding in Arbitrary Graphs", *SIAM Journal on Computing*, Vol. 20, No. 2, pp. 209–221, 1991.

64. Adams-Smith, D. J. and D. R. Shier, "Generating Random Test Networks for Shortest Path Algorithms", *Operations Research and Cyber-Infrastructure*, pp. 295–308,

2009.

65. Reinelt, G., "TSPLIB—A Traveling Salesman Problem Library", *ORSA Journal on Computing*, Vol. 3, No. 4, pp. 376–384, 1991.

66. Dolan, E. D. and J. J. Moré, "Benchmarking Optimization Software with Performance Profiles", *Mathematical Programming*, Vol. 91, No. 2, pp. 201–213, 2002.

67. Dezső, B., A. Jüttner and P. Kovács, "LEMON–An Open Source C++ Graph Template Library", *Electronic Notes in Theoretical Computer Science*, Vol. 264, No. 5, pp. 23–45, 2011.

68. Mannino, C. and A. Sassano, "Edge Projection and the Maximum Stable Set Problem", *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, Vol. 26, pp. 205–219, New Jersey, USA, 1996.

APPENDIX A: INSTANCES

In the computational experiments, 420 randomly generated instances and 420 realistic instances are used. Instance numbers from 1 to 420 are randomly generated instances and the remaining from 421 to 840 are the realistic ATSP instances [65]. Instance properties are reported in Table A.1. "Ins." column is the instance id number. "Best" column give the best known solution's objective value; if no solution is present "NoSol" is shown in this column. "Status" column is "Optimal" if that instance is solved to optimality by any of the algorithms, or "Feasible" if no solution is found or CPU time limit is reached and the optimality of the best known solution is not proven. "Seed" column simply give the random seed used at generation. "$|V(N)|$", "$|A(N)|$", and "$|E(C)|$" columns give the number of vertices in the network, the number of arcs in the network, and the number of edges in the conflict graph respectively. "Γ" and "Ω" columns give the network and conflict density parameters for each instance. Three instances are generated for each setting of the parameters with random seeds.

Table A.1. Test instance parameters.

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
1	719	Optimal	119	30	174	1384	0.2	0.1	2	1016	Optimal	1697	30	174	1384	0.2	0.1
3	917	Optimal	32733	30	174	1384	0.2	0.1	4	1515	Optimal	22804	30	174	2941	0.2	0.2
5	1335	Optimal	17231	30	174	2941	0.2	0.2	6	1065	Optimal	17856	30	174	2941	0.2	0.2
7	1365	Optimal	30949	30	174	4498	0.2	0.3	8	1620	Optimal	14924	30	174	4498	0.2	0.3
9	1665	Optimal	25867	30	174	4498	0.2	0.3	10	1515	Optimal	13479	30	174	5882	0.2	0.4
11	1350	Optimal	22933	30	174	5882	0.2	0.4	12	1215	Optimal	29137	30	174	5882	0.2	0.4
13	345	Optimal	10029	30	261	3380	0.3	0.1	14	431	Optimal	2815	30	261	3380	0.3	0.1
15	517	Optimal	3968	30	261	3380	0.3	0.1	16	1260	Optimal	14316	30	261	6760	0.3	0.2
17	1485	Optimal	28878	30	261	6760	0.3	0.2	18	1126	Optimal	9127	30	261	6760	0.3	0.2
19	1170	Optimal	32686	30	261	10140	0.3	0.3	20	1950	Optimal	23874	30	261	10140	0.3	0.3
21	1390	Optimal	30380	30	261	10140	0.3	0.3	22	1215	Optimal	24985	30	261	13520	0.3	0.4
23	1170	Optimal	25008	30	261	13520	0.3	0.4	24	1395	Optimal	7437	30	261	13520	0.3	0.4
25	347	Optimal	23223	30	348	5899	0.4	0.1	26	281	Optimal	16336	30	348	5899	0.4	0.1
27	360	Optimal	20270	30	348	5899	0.4	0.1	28	1320	Optimal	29548	30	348	11798	0.4	0.2
29	1149	Optimal	25955	30	348	11798	0.4	0.2	30	1575	Optimal	32203	30	348	11798	0.4	0.2
31	1095	Optimal	15120	30	348	18044	0.4	0.3	32	1335	Optimal	5745	30	348	18044	0.4	0.3
33	1020	Optimal	9843	30	348	18044	0.4	0.3	34	1560	Optimal	5384	30	348	23943	0.4	0.4
35	1080	Optimal	15581	30	348	23943	0.4	0.4	36	1665	Optimal	22147	30	348	23943	0.4	0.4
37	239	Optimal	483	30	435	9114	0.5	0.1	38	306	Optimal	8786	30	435	9114	0.5	0.1
39	296	Optimal	9812	30	435	9114	0.5	0.1	40	1157	Optimal	25275	30	435	18662	0.5	0.2
41	1005	Optimal	13905	30	435	18662	0.5	0.2	42	1095	Optimal	3828	30	435	18662	0.5	0.2

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
43	1575	Optimal	17129	30	435	28210	0.5	0.3	44	1044	Optimal	418	30	435	28210	0.5	0.3
45	1755	Optimal	7919	30	435	28210	0.5	0.3	46	1365	Optimal	28532	30	435	37758	0.5	0.4
47	1035	Optimal	3732	30	435	37758	0.5	0.4	48	1680	Optimal	6611	30	435	37758	0.5	0.4
49	197	Optimal	17155	30	522	13546	0.6	0.1	50	236	Optimal	15817	30	522	13546	0.6	0.1
51	258	Optimal	30575	30	522	13546	0.6	0.1	52	491	Optimal	15221	30	522	27092	0.6	0.2
53	584	Optimal	14084	30	522	27092	0.6	0.2	54	735	Feasible	26007	30	522	27092	0.6	0.2
55	1245	Optimal	15535	30	522	40638	0.6	0.3	56	1215	Optimal	15971	30	522	40638	0.6	0.3
57	1275	Optimal	26499	30	522	40638	0.6	0.3	58	1605	Optimal	28036	30	522	54184	0.6	0.4
59	1365	Optimal	15514	30	522	54184	0.6	0.4	60	1290	Optimal	17827	30	522	54184	0.6	0.4
61	167	Optimal	8430	30	609	18240	0.7	0.1	62	211	Optimal	15079	30	609	18240	0.7	0.1
63	159	Optimal	6061	30	609	18240	0.7	0.1	64	447	Optimal	19107	30	609	36480	0.7	0.2
65	658	Feasible	17048	30	609	36480	0.7	0.2	66	437	Optimal	22865	30	609	36480	0.7	0.2
67	1560	Optimal	155	30	609	55328	0.7	0.3	68	1425	Optimal	32042	30	609	55328	0.7	0.3
69	1680	Optimal	3647	30	609	55328	0.7	0.3	70	1455	Optimal	2205	30	609	73568	0.7	0.4
71	1080	Optimal	1808	30	609	73568	0.7	0.4	72	1425	Optimal	29339	30	609	73568	0.7	0.4
73	224	Optimal	26951	30	696	23630	0.8	0.1	74	164	Optimal	13883	30	696	23630	0.8	0.1
75	156	Optimal	24433	30	696	23630	0.8	0.1	76	515	Feasible	14826	30	696	47955	0.8	0.2
77	328	Optimal	20181	30	696	47955	0.8	0.2	78	403	Feasible	5869	30	696	47955	0.8	0.2
79	1605	Optimal	6669	30	696	72280	0.8	0.3	80	1560	Optimal	15763	30	696	72280	0.8	0.3
81	1185	Optimal	168	30	696	72280	0.8	0.3	82	1215	Optimal	17843	30	696	96605	0.8	0.4
83	1305	Optimal	20838	30	696	96605	0.8	0.4	84	1695	Optimal	31080	30	696	96605	0.8	0.4
85	976	Optimal	22530	40	312	4665	0.2	0.1	86	1038	Optimal	14881	40	312	4665	0.2	0.1
87	1247	Optimal	30570	40	312	4665	0.2	0.1	88	2130	Optimal	13570	40	312	9641	0.2	0.2
89	2325	Optimal	1059	40	312	9641	0.2	0.2	90	1545	Optimal	1900	40	312	9641	0.2	0.2
91	1875	Optimal	10028	40	312	14306	0.2	0.3	92	1980	Optimal	23460	40	312	14306	0.2	0.3
93	2175	Optimal	30065	40	312	14306	0.2	0.3	94	1755	Optimal	9823	40	312	19282	0.2	0.4
95	1770	Optimal	12666	40	312	19282	0.2	0.4	96	1350	Optimal	4769	40	312	19282	0.2	0.4
97	473	Optimal	5155	40	468	10741	0.3	0.1	98	624	Feasible	12080	40	468	10741	0.3	0.1
99	701	Feasible	3877	40	468	10741	0.3	0.1	100	1489	Optimal	874	40	468	21482	0.3	0.2
101	1935	Optimal	11479	40	468	21482	0.3	0.2	102	2010	Optimal	16315	40	468	21482	0.3	0.2
103	1755	Optimal	23500	40	468	32690	0.3	0.3	104	2595	Optimal	13299	40	468	32690	0.3	0.3
105	2070	Optimal	8017	40	468	32690	0.3	0.3	106	2115	Optimal	12338	40	468	43431	0.3	0.4
107	2190	Optimal	22810	40	468	43431	0.3	0.4	108	1680	Optimal	31813	40	468	43431	0.3	0.4
109	482	Feasible	27081	40	624	19313	0.4	0.1	110	577	Feasible	11175	40	624	19313	0.4	0.1
111	401	Feasible	5280	40	624	19313	0.4	0.1	112	1527	Feasible	27226	40	624	38626	0.4	0.2
113	NoSol	Feasible	21822	40	624	38626	0.4	0.2	114	1890	Feasible	8104	40	624	38626	0.4	0.2
115	2295	Optimal	11158	40	624	57939	0.4	0.3	116	1770	Optimal	29005	40	624	57939	0.4	0.3
117	2025	Optimal	5732	40	624	57939	0.4	0.3	118	2130	Optimal	32330	40	624	77252	0.4	0.4
119	1710	Optimal	27587	40	624	77252	0.4	0.4	120	2415	Optimal	4862	40	624	77252	0.4	0.4
121	373	Feasible	25964	40	780	30381	0.5	0.1	122	409	Feasible	32551	40	780	30381	0.5	0.1
123	390	Feasible	3980	40	780	30381	0.5	0.1	124	NoSol	Feasible	27386	40	780	60762	0.5	0.2
125	NoSol	Feasible	21405	40	780	60762	0.5	0.2	126	NoSol	Feasible	26845	40	780	60762	0.5	0.2
127	2415	Optimal	24418	40	780	91143	0.5	0.3	128	1860	Optimal	306	40	780	91143	0.5	0.3
129	1965	Optimal	31401	40	780	91143	0.5	0.3	130	1830	Optimal	4386	40	780	121524	0.5	0.4
131	2430	Optimal	30961	40	780	121524	0.5	0.4	132	1860	Optimal	9934	40	780	121524	0.5	0.4
133	308	Feasible	13411	40	936	43010	0.6	0.1	134	244	Feasible	15553	40	936	43010	0.6	0.1
135	266	Optimal	24134	40	936	43010	0.6	0.1	136	NoSol	Feasible	12497	40	936	86955	0.6	0.2
137	NoSol	Feasible	13640	40	936	86955	0.6	0.2	138	NoSol	Feasible	20690	40	936	86955	0.6	0.2
139	1875	Optimal	6632	40	936	130900	0.6	0.3	140	2115	Optimal	22615	40	936	130900	0.6	0.3
141	1740	Optimal	9861	40	936	130900	0.6	0.3	142	1875	Optimal	20654	40	936	174845	0.6	0.4
143	1785	Optimal	19021	40	936	174845	0.6	0.4	144	2160	Optimal	23520	40	936	174845	0.6	0.4
145	289	Feasible	28329	40	1092	58914	0.7	0.1	146	332	Optimal	18681	40	1092	58914	0.7	0.1
147	242	Feasible	20171	40	1092	58914	0.7	0.1	148	NoSol	Feasible	17536	40	1092	118919	0.7	0.2
149	NoSol	Feasible	17822	40	1092	118919	0.7	0.2	150	NoSol	Feasible	22729	40	1092	118919	0.7	0.2
151	NoSol	Feasible	3992	40	1092	177833	0.7	0.3	152	2490	Optimal	1826	40	1092	177833	0.7	0.3
153	2580	Feasible	28990	40	1092	177833	0.7	0.3	154	1470	Optimal	29864	40	1092	237838	0.7	0.4
155	1965	Optimal	2655	40	1092	237838	0.7	0.4	156	2220	Optimal	4828	40	1092	237838	0.7	0.4
157	239	Optimal	11744	40	1248	77314	0.8	0.1	158	238	Optimal	23090	40	1248	77314	0.8	0.1
159	309	Feasible	9556	40	1248	77314	0.8	0.1	160	NoSol	Feasible	22192	40	1248	154628	0.8	0.2

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
161	NoSol	Feasible	15111	40	1248	154628	0.8	0.2	162	NoSol	Feasible	24531	40	1248	154628	0.8	0.2
163	NoSol	Feasible	30150	40	1248	233189	0.8	0.3	164	NoSol	Feasible	385	40	1248	233189	0.8	0.3
165	NoSol	Feasible	7730	40	1248	233189	0.8	0.3	166	1875	Optimal	18213	40	1248	310503	0.8	0.4
167	2415	Optimal	20013	40	1248	310503	0.8	0.4	168	1830	Optimal	20032	40	1248	310503	0.8	0.4
169	1260	Feasible	5526	50	490	11736	0.2	0.1	170	1749	Feasible	32089	50	490	11736	0.2	0.1
171	1543	Feasible	32427	50	490	11736	0.2	0.1	172	2445	Optimal	14153	50	490	23961	0.2	0.2
173	2775	Optimal	15139	50	490	23961	0.2	0.2	174	2361	Optimal	15779	50	490	23961	0.2	0.2
175	3060	Optimal	27914	50	490	35697	0.2	0.3	176	3870	Optimal	12253	50	490	35697	0.2	0.3
177	2235	Optimal	25301	50	490	35697	0.2	0.3	178	2385	Optimal	30059	50	490	47922	0.2	0.4
179	2220	Optimal	2781	50	490	47922	0.2	0.4	180	1890	Optimal	3013	50	490	47922	0.2	0.4
181	993	Feasible	12488	50	735	26424	0.3	0.1	182	1023	Feasible	2760	50	735	26424	0.3	0.1
183	698	Feasible	8767	50	735	26424	0.3	0.1	184	2820	Optimal	19684	50	735	53582	0.3	0.2
185	3210	Optimal	2829	50	735	53582	0.3	0.2	186	2731	Feasible	4424	50	735	53582	0.3	0.2
187	2415	Optimal	27367	50	735	80740	0.3	0.3	188	2265	Optimal	21392	50	735	80740	0.3	0.3
189	2985	Optimal	31046	50	735	80740	0.3	0.3	190	2325	Optimal	25350	50	735	107898	0.3	0.4
191	2415	Optimal	26711	50	735	107898	0.3	0.4	192	2940	Optimal	8375	50	735	107898	0.3	0.4
193	500	Feasible	18596	50	980	47971	0.4	0.1	194	769	Feasible	13387	50	980	47971	0.4	0.1
195	721	Feasible	6952	50	980	47971	0.4	0.1	196	NoSol	Feasible	8324	50	980	95942	0.4	0.2
197	NoSol	Feasible	15029	50	980	95942	0.4	0.2	198	NoSol	Feasible	29906	50	980	95942	0.4	0.2
199	2895	Optimal	26282	50	980	143913	0.4	0.3	200	2190	Optimal	31898	50	980	143913	0.4	0.3
201	2595	Optimal	403	50	980	143913	0.4	0.3	202	2415	Optimal	13081	50	980	191884	0.4	0.4
203	2715	Optimal	26776	50	980	191884	0.4	0.4	204	3750	Optimal	8617	50	980	191884	0.4	0.4
205	487	Feasible	6291	50	1225	74664	0.5	0.1	206	483	Feasible	16806	50	1225	74664	0.5	0.1
207	755	Feasible	670	50	1225	74664	0.5	0.1	208	NoSol	Feasible	20864	50	1225	149328	0.5	0.2
209	NoSol	Feasible	18171	50	1225	149328	0.5	0.2	210	NoSol	Feasible	6717	50	1225	149328	0.5	0.2
211	2430	Optimal	19861	50	1225	223992	0.5	0.3	212	2145	Optimal	7265	50	1225	223992	0.5	0.3
213	2205	Optimal	9728	50	1225	223992	0.5	0.3	214	2640	Optimal	31731	50	1225	299880	0.5	0.4
215	2655	Optimal	21919	50	1225	299880	0.5	0.4	216	2580	Optimal	25472	50	1225	299880	0.5	0.4
217	403	Feasible	887	50	1470	107237	0.6	0.1	218	549	Feasible	23167	50	1470	107237	0.6	0.1
219	417	Feasible	7439	50	1470	107237	0.6	0.1	220	NoSol	Feasible	1476	50	1470	215943	0.6	0.2
221	NoSol	Feasible	26823	50	1470	215943	0.6	0.2	222	NoSol	Feasible	6992	50	1470	215943	0.6	0.2
223	NoSol	Feasible	573	50	1470	323180	0.6	0.3	224	NoSol	Feasible	6148	50	1470	323180	0.6	0.3
225	NoSol	Feasible	8108	50	1470	323180	0.6	0.3	226	3435	Optimal	22754	50	1470	431886	0.6	0.4
227	2280	Optimal	27789	50	1470	431886	0.6	0.4	228	2745	Optimal	8026	50	1470	431886	0.6	0.4
229	407	Feasible	11825	50	1715	145690	0.7	0.1	230	618	Feasible	12207	50	1715	145690	0.7	0.1
231	454	Feasible	30670	50	1715	145690	0.7	0.1	232	NoSol	Feasible	10810	50	1715	293094	0.7	0.2
233	NoSol	Feasible	30159	50	1715	293094	0.7	0.2	234	NoSol	Feasible	6114	50	1715	293094	0.7	0.2
235	NoSol	Feasible	27441	50	1715	440498	0.7	0.3	236	NoSol	Feasible	13852	50	1715	440498	0.7	0.3
237	NoSol	Feasible	16343	50	1715	440498	0.7	0.3	238	2565	Optimal	20044	50	1715	587902	0.7	0.4
239	2565	Optimal	32464	50	1715	587902	0.7	0.4	240	2910	Optimal	94	50	1715	587902	0.7	0.4
241	474	Feasible	9063	50	1960	191982	0.8	0.1	242	406	Feasible	32047	50	1960	191982	0.8	0.1
243	325	Feasible	3621	50	1960	191982	0.8	0.1	244	NoSol	Feasible	6018	50	1960	383964	0.8	0.2
245	NoSol	Feasible	14307	50	1960	383964	0.8	0.2	246	NoSol	Feasible	15746	50	1960	383964	0.8	0.2
247	NoSol	Feasible	21724	50	1960	575946	0.8	0.3	248	NoSol	Feasible	25571	50	1960	575946	0.8	0.3
249	NoSol	Feasible	438	50	1960	575946	0.8	0.3	250	2520	Optimal	13307	50	1960	767928	0.8	0.4
251	2490	Optimal	4425	50	1960	767928	0.8	0.4	252	2805	Optimal	5016	50	1960	767928	0.8	0.4
253	NoSol	Feasible	25732	60	708	24745	0.2	0.1	254	NoSol	Feasible	25964	60	708	24745	0.2	0.1
255	NoSol	Feasible	29955	60	708	24745	0.2	0.1	256	2925	Optimal	28393	60	708	49490	0.2	0.2
257	3289	Optimal	11558	60	708	49490	0.2	0.2	258	3405	Optimal	17991	60	708	49490	0.2	0.2
259	2970	Optimal	10925	60	708	74942	0.2	0.3	260	3435	Optimal	14435	60	708	74942	0.2	0.3
261	2970	Optimal	15943	60	708	74942	0.2	0.3	262	3735	Optimal	6585	60	708	99687	0.2	0.4
263	4005	Optimal	31077	60	708	99687	0.2	0.4	264	3540	Optimal	6627	60	708	99687	0.2	0.4
265	NoSol	Feasible	3072	60	1062	56233	0.3	0.1	266	NoSol	Feasible	5455	60	1062	56233	0.3	0.1
267	NoSol	Feasible	7666	60	1062	56233	0.3	0.1	268	NoSol	Feasible	17305	60	1062	112466	0.3	0.2
269	NoSol	Feasible	3025	60	1062	112466	0.3	0.2	270	NoSol	Feasible	5132	60	1062	112466	0.3	0.2
271	2445	Optimal	19317	60	1062	168699	0.3	0.3	272	3360	Optimal	11646	60	1062	168699	0.3	0.3
273	3495	Optimal	2632	60	1062	168699	0.3	0.3	274	3540	Optimal	23069	60	1062	224932	0.3	0.4
275	3285	Optimal	23024	60	1062	224932	0.3	0.4	276	2580	Optimal	23055	60	1062	224932	0.3	0.4
277	NoSol	Feasible	25825	60	1416	99050	0.4	0.1	278	NoSol	Feasible	12547	60	1416	99050	0.4	0.1

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
279	NoSol	Feasible	12629	60	1416	99050	0.4	0.1	280	NoSol	Feasible	25613	60	1416	199515	0.4	0.2
281	NoSol	Feasible	12015	60	1416	199515	0.4	0.2	282	NoSol	Feasible	17069	60	1416	199515	0.4	0.2
283	2805	Optimal	32016	60	1416	299980	0.4	0.3	284	2730	Optimal	17333	60	1416	299980	0.4	0.3
285	3135	Optimal	2415	60	1416	299980	0.4	0.3	286	2595	Optimal	20388	60	1416	400445	0.4	0.4
287	2775	Optimal	14570	60	1416	400445	0.4	0.4	288	3375	Optimal	14865	60	1416	400445	0.4	0.4
289	NoSol	Feasible	7042	60	1770	155672	0.5	0.1	290	NoSol	Feasible	1934	60	1770	155672	0.5	0.1
291	1119	Feasible	456	60	1770	155672	0.5	0.1	292	NoSol	Feasible	22472	60	1770	313113	0.5	0.2
293	NoSol	Feasible	18842	60	1770	313113	0.5	0.2	294	NoSol	Feasible	21460	60	1770	313113	0.5	0.2
295	NoSol	Feasible	17223	60	1770	468785	0.5	0.3	296	3270	Feasible	16515	60	1770	468785	0.5	0.3
297	3315	Feasible	1809	60	1770	468785	0.5	0.3	298	3315	Optimal	24579	60	1770	626226	0.5	0.4
299	3360	Optimal	16772	60	1770	626226	0.5	0.4	300	2805	Optimal	3080	60	1770	626226	0.5	0.4
301	NoSol	Feasible	27539	60	2124	225038	0.6	0.1	302	NoSol	Feasible	4689	60	2124	225038	0.6	0.1
303	NoSol	Feasible	5029	60	2124	225038	0.6	0.1	304	NoSol	Feasible	25927	60	2124	450076	0.6	0.2
305	NoSol	Feasible	16908	60	2124	450076	0.6	0.2	306	NoSol	Feasible	4832	60	2124	450076	0.6	0.2
307	NoSol	Feasible	7669	60	2124	675114	0.6	0.3	308	NoSol	Feasible	26816	60	2124	675114	0.6	0.3
309	NoSol	Feasible	29375	60	2124	675114	0.6	0.3	310	2940	Optimal	20924	60	2124	900152	0.6	0.4
311	3435	Optimal	24532	60	2124	900152	0.6	0.4	312	3480	Optimal	24963	60	2124	900152	0.6	0.4
313	NoSol	Feasible	28513	60	2478	304671	0.7	0.1	314	NoSol	Feasible	26603	60	2478	304671	0.7	0.1
315	NoSol	Feasible	4387	60	2478	304671	0.7	0.1	316	NoSol	Feasible	13364	60	2478	611819	0.7	0.2
317	NoSol	Feasible	26504	60	2478	611819	0.7	0.2	318	NoSol	Feasible	32254	60	2478	611819	0.7	0.2
319	NoSol	Feasible	29289	60	2478	918967	0.7	0.3	320	NoSol	Feasible	22311	60	2478	918967	0.7	0.3
321	NoSol	Feasible	8379	60	2478	918967	0.7	0.3	322	2430	Feasible	8176	60	2478	1226115	0.7	0.4
323	2835	Optimal	1218	60	2478	1226115	0.7	0.4	324	3450	Optimal	5783	60	2478	1226115	0.7	0.4
325	NoSol	Feasible	29783	60	2832	399171	0.8	0.1	326	NoSol	Feasible	26299	60	2832	399171	0.8	0.1
327	782	Feasible	26317	60	2832	399171	0.8	0.1	328	NoSol	Feasible	24540	60	2832	801173	0.8	0.2
329	NoSol	Feasible	20523	60	2832	801173	0.8	0.2	330	NoSol	Feasible	26461	60	2832	801173	0.8	0.2
331	NoSol	Feasible	9228	60	2832	1200344	0.8	0.3	332	NoSol	Feasible	14769	60	2832	1200344	0.8	0.3
333	NoSol	Feasible	935	60	2832	1200344	0.8	0.3	334	NoSol	Feasible	16500	60	2832	1602346	0.8	0.4
335	3000	Feasible	2276	60	2832	1602346	0.8	0.4	336	NoSol	Feasible	7684	60	2832	1602346	0.8	0.4
337	NoSol	Feasible	30917	70	966	46320	0.2	0.1	338	NoSol	Feasible	17807	70	966	46320	0.2	0.1
339	NoSol	Feasible	23003	70	966	46320	0.2	0.1	340	3255	Feasible	3555	70	966	92640	0.2	0.2
341	3135	Optimal	12272	70	966	92640	0.2	0.2	342	3840	Optimal	32036	70	966	92640	0.2	0.2
343	3585	Optimal	8259	70	966	138960	0.2	0.3	344	5970	Optimal	18948	70	966	138960	0.2	0.3
345	3390	Optimal	2826	70	966	138960	0.2	0.3	346	4020	Optimal	4135	70	966	186245	0.2	0.4
347	5805	Optimal	22160	70	966	186245	0.2	0.4	348	3420	Optimal	26146	70	966	186245	0.2	0.4
349	NoSol	Feasible	14049	70	1449	104256	0.3	0.1	350	NoSol	Feasible	11161	70	1449	104256	0.3	0.1
351	NoSol	Feasible	30515	70	1449	104256	0.3	0.1	352	NoSol	Feasible	23862	70	1449	208512	0.3	0.2
353	NoSol	Feasible	15157	70	1449	208512	0.3	0.2	354	NoSol	Feasible	9709	70	1449	208512	0.3	0.2
355	4545	Optimal	24983	70	1449	314216	0.3	0.3	356	4365	Optimal	10193	70	1449	314216	0.3	0.3
357	3390	Optimal	32765	70	1449	314216	0.3	0.3	358	4005	Optimal	22914	70	1449	418472	0.3	0.4
359	3405	Optimal	16441	70	1449	418472	0.3	0.4	360	4200	Optimal	29059	70	1449	418472	0.3	0.4
361	NoSol	Feasible	24844	70	1932	185376	0.4	0.1	362	NoSol	Feasible	12178	70	1932	185376	0.4	0.1
363	NoSol	Feasible	31571	70	1932	185376	0.4	0.1	364	NoSol	Feasible	16706	70	1932	372683	0.4	0.2
365	NoSol	Feasible	21302	70	1932	372683	0.4	0.2	366	NoSol	Feasible	29519	70	1932	372683	0.4	0.2
367	3090	Feasible	11796	70	1932	558059	0.4	0.3	368	NoSol	Feasible	20523	70	1932	558059	0.4	0.3
369	3825	Optimal	9419	70	1932	558059	0.4	0.3	370	3990	Optimal	10608	70	1932	745366	0.4	0.4
371	4695	Optimal	15676	70	1932	745366	0.4	0.4	372	4470	Optimal	26921	70	1932	745366	0.4	0.4
373	NoSol	Feasible	29526	70	2415	289680	0.5	0.1	374	NoSol	Feasible	21482	70	2415	289680	0.5	0.1
375	NoSol	Feasible	7445	70	2415	289680	0.5	0.1	376	NoSol	Feasible	7524	70	2415	581774	0.5	0.2
377	NoSol	Feasible	2705	70	2415	581774	0.5	0.2	378	NoSol	Feasible	7252	70	2415	581774	0.5	0.2
379	NoSol	Feasible	5358	70	2415	873868	0.5	0.3	380	4575	Feasible	12372	70	2415	873868	0.5	0.3
381	NoSol	Feasible	6444	70	2415	873868	0.5	0.3	382	3900	Optimal	26208	70	2415	1165962	0.5	0.4
383	4980	Optimal	11791	70	2415	1165962	0.5	0.4	384	4350	Optimal	1876	70	2415	1165962	0.5	0.4
385	NoSol	Feasible	15939	70	2898	417168	0.6	0.1	386	NoSol	Feasible	23120	70	2898	417168	0.6	0.1
387	NoSol	Feasible	24878	70	2898	417168	0.6	0.1	388	NoSol	Feasible	13472	70	2898	837233	0.6	0.2
389	NoSol	Feasible	29639	70	2898	837233	0.6	0.2	390	NoSol	Feasible	3583	70	2898	837233	0.6	0.2
391	NoSol	Feasible	28523	70	2898	1257298	0.6	0.3	392	NoSol	Feasible	27403	70	2898	1257298	0.6	0.3
393	NoSol	Feasible	27630	70	2898	1257298	0.6	0.3	394	3405	Optimal	23961	70	2898	1677363	0.6	0.4
395	5085	Optimal	1770	70	2898	1677363	0.6	0.4	396	4470	Optimal	31703	70	2898	1677363	0.6	0.4

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
397	NoSol	Feasible	3792	70	3381	571220	0.7	0.1	398	NoSol	Feasible	18146	70	3381	571220	0.7	0.1
399	NoSol	Feasible	7677	70	3381	571220	0.7	0.1	400	NoSol	Feasible	16789	70	3381	1142440	0.7	0.2
401	NoSol	Feasible	15511	70	3381	1142440	0.7	0.2	402	NoSol	Feasible	3818	70	3381	1142440	0.7	0.2
403	NoSol	Feasible	26876	70	3381	1713660	0.7	0.3	404	NoSol	Feasible	18598	70	3381	1713660	0.7	0.3
405	NoSol	Feasible	25060	70	3381	1713660	0.7	0.3	406	NoSol	Feasible	15054	70	3381	2284880	0.7	0.4
407	NoSol	Feasible	10204	70	3381	2284880	0.7	0.4	408	NoSol	Feasible	780	70	3381	2284880	0.7	0.4
409	NoSol	Feasible	6065	70	3864	745559	0.8	0.1	410	NoSol	Feasible	16144	70	3864	745559	0.8	0.1
411	NoSol	Feasible	932	70	3864	745559	0.8	0.1	412	NoSol	Feasible	19120	70	3864	1491118	0.8	0.2
413	NoSol	Feasible	14881	70	3864	1491118	0.8	0.2	414	NoSol	Feasible	21030	70	3864	1491118	0.8	0.2
415	NoSol	Feasible	21165	70	3864	2236677	0.8	0.3	416	NoSol	Feasible	18711	70	3864	2236677	0.8	0.3
417	NoSol	Feasible	31452	70	3864	2236677	0.8	0.3	418	NoSol	Feasible	7667	70	3864	2982236	0.8	0.4
419	NoSol	Feasible	30021	70	3864	2982236	0.8	0.4	420	NoSol	Feasible	7247	70	3864	2982236	0.8	0.4
421	15203	Optimal	12078	34	224	2453	0.2	0.1	422	10015	Optimal	20228	34	224	2453	0.2	0.1
423	8976	Optimal	24720	34	224	2453	0.2	0.1	424	10707	Optimal	21230	34	224	4906	0.2	0.2
425	11710	Optimal	30061	34	224	4906	0.2	0.2	426	12065	Optimal	30246	34	224	4906	0.2	0.2
427	7298	Optimal	8844	34	224	7359	0.2	0.3	428	12483	Optimal	11056	34	224	7359	0.2	0.3
429	15942	Optimal	22234	34	224	7359	0.2	0.3	430	15789	Optimal	11205	34	224	9812	0.2	0.4
431	14569	Optimal	13130	34	224	9812	0.2	0.4	432	12254	Optimal	32219	34	224	9812	0.2	0.4
433	9089	Optimal	11572	34	337	5376	0.3	0.1	434	5724	Optimal	19343	34	337	5376	0.3	0.1
435	7071	Optimal	27540	34	337	5376	0.3	0.1	436	11467	Optimal	1911	34	337	11088	0.3	0.2
437	8896	Optimal	28298	34	337	11088	0.3	0.2	438	10486	Optimal	31705	34	337	11088	0.3	0.2
439	15683	Optimal	16968	34	337	16800	0.3	0.3	440	9621	Optimal	23273	34	337	16800	0.3	0.3
441	12553	Optimal	31652	34	337	16800	0.3	0.3	442	11300	Optimal	3677	34	337	22512	0.3	0.4
443	15692	Optimal	4136	34	337	22512	0.3	0.4	444	9550	Optimal	17441	34	337	22512	0.3	0.4
445	4841	Optimal	3520	34	449	9856	0.4	0.1	446	5553	Optimal	13053	34	449	9856	0.4	0.1
447	4887	Optimal	26659	34	449	9856	0.4	0.1	448	11275	Optimal	13699	34	449	19712	0.4	0.2
449	18389	Optimal	27887	34	449	19712	0.4	0.2	450	9702	Optimal	30913	34	449	19712	0.4	0.2
451	12382	Optimal	19	34	449	30016	0.4	0.3	452	10856	Optimal	6669	34	449	30016	0.4	0.3
453	8944	Optimal	19405	34	449	30016	0.4	0.3	454	19163	Optimal	6248	34	449	39872	0.4	0.4
455	11900	Optimal	19637	34	449	39872	0.4	0.4	456	19822	Optimal	30222	34	449	39872	0.4	0.4
457	4884	Optimal	373	34	561	15680	0.5	0.1	458	4910	Optimal	3479	34	561	15680	0.5	0.1
459	4661	Optimal	8609	34	561	15680	0.5	0.1	460	7239	Optimal	2298	34	561	31360	0.5	0.2
461	19910	Optimal	6834	34	561	31360	0.5	0.2	462	15033	Optimal	9129	34	561	31360	0.5	0.2
463	10522	Optimal	2695	34	561	47040	0.5	0.3	464	17236	Optimal	10178	34	561	47040	0.5	0.3
465	16187	Optimal	21715	34	561	47040	0.5	0.3	466	11634	Optimal	22950	34	561	62720	0.5	0.4
467	10914	Optimal	23915	34	561	62720	0.5	0.4	468	11024	Optimal	32112	34	561	62720	0.5	0.4
469	4087	Feasible	2608	34	673	22176	0.6	0.1	470	3979	Optimal	4916	34	673	22176	0.6	0.1
471	4219	Optimal	15990	34	673	22176	0.6	0.1	472	NoSol	Feasible	20407	34	673	45024	0.6	0.2
473	NoSol	Feasible	32165	34	673	45024	0.6	0.2	474	NoSol	Feasible	32604	34	673	45024	0.6	0.2
475	17189	Optimal	2773	34	673	67200	0.6	0.3	476	11580	Optimal	12459	34	673	67200	0.6	0.3
477	9697	Optimal	22838	34	673	67200	0.6	0.3	478	15837	Optimal	21498	34	673	90048	0.6	0.4
479	16861	Optimal	28155	34	673	90048	0.6	0.4	480	12961	Optimal	29342	34	673	90048	0.6	0.4
481	3771	Optimal	9112	34	785	30576	0.7	0.1	482	3869	Optimal	10292	34	785	30576	0.7	0.1
483	3419	Optimal	13087	34	785	30576	0.7	0.1	484	NoSol	Feasible	10245	34	785	61152	0.7	0.2
485	NoSol	Feasible	24812	34	785	61152	0.7	0.2	486	NoSol	Feasible	31962	34	785	61152	0.7	0.2
487	8094	Optimal	33	34	785	91728	0.7	0.3	488	7818	Optimal	9116	34	785	91728	0.7	0.3
489	11857	Optimal	21911	34	785	91728	0.7	0.3	490	12343	Optimal	19486	34	785	123088	0.7	0.4
491	10120	Optimal	25933	34	785	123088	0.7	0.4	492	13695	Optimal	27698	34	785	123088	0.7	0.4
493	3482	Optimal	7826	34	898	39468	0.8	0.1	494	3534	Optimal	13418	34	898	39468	0.8	0.1
495	3708	Feasible	18080	34	898	39468	0.8	0.1	496	NoSol	Feasible	21640	34	898	79833	0.8	0.2
497	NoSol	Feasible	26745	34	898	79833	0.8	0.2	498	NoSol	Feasible	29550	34	898	79833	0.8	0.2
499	18371	Optimal	5980	34	898	120198	0.8	0.3	500	11892	Optimal	9712	34	898	120198	0.8	0.3
501	9333	Optimal	31229	34	898	120198	0.8	0.3	502	12505	Optimal	5597	34	898	160563	0.8	0.4
503	11695	Optimal	12174	34	898	160563	0.8	0.4	504	9471	Optimal	23649	34	898	160563	0.8	0.4
505	16348	Optimal	1352	45	396	7505	0.2	0.1	506	27810	Feasible	29856	45	396	7505	0.2	0.1
507	22051	Optimal	32296	45	396	7505	0.2	0.1	508	19279	Optimal	16346	45	396	15405	0.2	0.2
509	15027	Optimal	19476	45	396	15405	0.2	0.2	510	14494	Optimal	20113	45	396	15405	0.2	0.2
511	18053	Optimal	2631	45	396	23305	0.2	0.3	512	13505	Optimal	11912	45	396	23305	0.2	0.3
513	16651	Optimal	28831	45	396	23305	0.2	0.3	514	18747	Optimal	7030	45	396	31205	0.2	0.4

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
515	17855	Optimal	7332	45	396	31205	0.2	0.4	516	26307	Optimal	12870	45	396	31205	0.2	0.4
517	15998	Feasible	11815	45	594	17197	0.3	0.1	518	21237	Feasible	16764	45	594	17197	0.3	0.1
519	25460	Feasible	17174	45	594	17197	0.3	0.1	520	17304	Optimal	7093	45	594	34987	0.3	0.2
521	26405	Optimal	14689	45	594	34987	0.3	0.2	522	20264	Optimal	26541	45	594	34987	0.3	0.2
523	19128	Optimal	7749	45	594	52777	0.3	0.3	524	21163	Optimal	16804	45	594	52777	0.3	0.3
525	13952	Optimal	24705	45	594	52777	0.3	0.3	526	18035	Optimal	7926	45	594	69974	0.3	0.4
527	27193	Optimal	15753	45	594	69974	0.3	0.4	528	14341	Optimal	16166	45	594	69974	0.3	0.4
529	12804	Feasible	4358	45	792	30849	0.4	0.1	530	13902	Feasible	15442	45	792	30849	0.4	0.1
531	14048	Feasible	24821	45	792	30849	0.4	0.1	532	14178	Optimal	2460	45	792	62489	0.4	0.2
533	20577	Feasible	13933	45	792	62489	0.4	0.2	534	14830	Optimal	31049	45	792	62489	0.4	0.2
535	16045	Optimal	251	45	792	93338	0.4	0.3	536	22226	Optimal	658	45	792	93338	0.4	0.3
537	24899	Optimal	2009	45	792	93338	0.4	0.3	538	18448	Optimal	3431	45	792	124978	0.4	0.4
539	13831	Optimal	3548	45	792	124978	0.4	0.4	540	17335	Optimal	22177	45	792	124978	0.4	0.4
541	13289	Feasible	9536	45	990	48461	0.5	0.1	542	13114	Feasible	14075	45	990	48461	0.5	0.1
543	11007	Feasible	18770	45	990	48461	0.5	0.1	544	NoSol	Feasible	7054	45	990	97911	0.5	0.2
545	NoSol	Feasible	15471	45	990	97911	0.5	0.2	546	NoSol	Feasible	21380	45	990	97911	0.5	0.2
547	17443	Optimal	3999	45	990	146372	0.5	0.3	548	23058	Optimal	27408	45	990	146372	0.5	0.3
549	24689	Optimal	29748	45	990	146372	0.5	0.3	550	18276	Optimal	17159	45	990	195822	0.5	0.4
551	21556	Optimal	26144	45	990	195822	0.5	0.4	552	18300	Optimal	30223	45	990	195822	0.5	0.4
553	8763	Feasible	10494	45	1188	70033	0.6	0.1	554	9868	Feasible	12803	45	1188	70033	0.6	0.1
555	9142	Feasible	17471	45	1188	70033	0.6	0.1	556	NoSol	Feasible	19147	45	1188	140066	0.6	0.2
557	NoSol	Feasible	26599	45	1188	140066	0.6	0.2	558	NoSol	Feasible	28928	45	1188	140066	0.6	0.2
559	15460	Optimal	2870	45	1188	211286	0.6	0.3	560	18239	Optimal	14783	45	1188	211286	0.6	0.3
561	16964	Optimal	16730	45	1188	211286	0.6	0.3	562	16636	Optimal	6045	45	1188	281319	0.6	0.4
563	17875	Optimal	17606	45	1188	281319	0.6	0.4	564	20584	Optimal	32700	45	1188	281319	0.6	0.4
565	8624	Feasible	20645	45	1386	95565	0.7	0.1	566	7877	Feasible	22204	45	1386	95565	0.7	0.1
567	6874	Feasible	32672	45	1386	95565	0.7	0.1	568	NoSol	Feasible	8049	45	1386	191130	0.7	0.2
569	NoSol	Feasible	9240	45	1386	191130	0.7	0.2	570	NoSol	Feasible	32095	45	1386	191130	0.7	0.2
571	17148	Feasible	1582	45	1386	286695	0.7	0.3	572	NoSol	Feasible	2688	45	1386	286695	0.7	0.3
573	23267	Feasible	27326	45	1386	286695	0.7	0.3	574	25706	Optimal	6387	45	1386	383645	0.7	0.4
575	21418	Optimal	11809	45	1386	383645	0.7	0.4	576	15978	Optimal	24100	45	1386	383645	0.7	0.4
577	7262	Feasible	5273	45	1584	125057	0.8	0.1	578	6996	Feasible	12612	45	1584	125057	0.8	0.1
579	7889	Feasible	29310	45	1584	125057	0.8	0.1	580	NoSol	Feasible	11717	45	1584	250114	0.8	0.2
581	NoSol	Feasible	18827	45	1584	250114	0.8	0.2	582	NoSol	Feasible	31784	45	1584	250114	0.8	0.2
583	NoSol	Feasible	19488	45	1584	375171	0.8	0.3	584	NoSol	Feasible	24409	45	1584	375171	0.8	0.3
585	NoSol	Feasible	30359	45	1584	375171	0.8	0.3	586	19668	Optimal	819	45	1584	500228	0.8	0.4
587	17266	Optimal	13251	45	1584	500228	0.8	0.4	588	18415	Optimal	29972	45	1584	500228	0.8	0.4
589	NoSol	Feasible	4293	56	616	18450	0.2	0.1	590	NoSol	Feasible	11645	56	616	18450	0.2	0.1
591	23480	Feasible	28581	56	616	18450	0.2	0.1	592	25335	Optimal	1998	56	616	37515	0.2	0.2
593	18650	Optimal	20715	56	616	37515	0.2	0.2	594	21187	Optimal	31597	56	616	37515	0.2	0.2
595	20561	Optimal	2777	56	616	56580	0.2	0.3	596	36818	Optimal	16022	56	616	56580	0.2	0.3
597	24493	Optimal	31350	56	616	56580	0.2	0.3	598	18807	Optimal	21521	56	616	75645	0.2	0.4
599	21979	Optimal	21926	56	616	75645	0.2	0.4	600	19301	Optimal	23436	56	616	75645	0.2	0.4
601	NoSol	Feasible	1100	56	924	42458	0.3	0.1	602	NoSol	Feasible	16877	56	924	42458	0.3	0.1
603	NoSol	Feasible	22871	56	924	42458	0.3	0.1	604	21716	Optimal	4973	56	924	84916	0.3	0.2
605	22611	Optimal	12404	56	924	84916	0.3	0.2	606	26994	Optimal	21009	56	924	84916	0.3	0.2
607	25600	Optimal	3214	56	924	127374	0.3	0.3	608	26986	Optimal	7093	56	924	127374	0.3	0.3
609	21232	Optimal	15181	56	924	127374	0.3	0.3	610	32580	Optimal	8927	56	924	169832	0.3	0.4
611	22638	Optimal	11383	56	924	169832	0.3	0.4	612	23912	Optimal	19551	56	924	169832	0.3	0.4
613	NoSol	Feasible	2936	56	1232	75091	0.4	0.1	614	NoSol	Feasible	17302	56	1232	75091	0.4	0.1
615	NoSol	Feasible	23954	56	1232	75091	0.4	0.1	616	NoSol	Feasible	5484	56	1232	151413	0.4	0.2
617	NoSol	Feasible	16268	56	1232	151413	0.4	0.2	618	NoSol	Feasible	25329	56	1232	151413	0.4	0.2
619	43412	Optimal	7017	56	1232	226504	0.4	0.3	620	37484	Optimal	13594	56	1232	226504	0.4	0.3
621	26521	Optimal	21826	56	1232	226504	0.4	0.3	622	24385	Optimal	1764	56	1232	302826	0.4	0.4
623	27302	Optimal	2039	56	1232	302826	0.4	0.4	624	24523	Optimal	7524	56	1232	302826	0.4	0.4
625	NoSol	Feasible	8622	56	1540	118503	0.5	0.1	626	NoSol	Feasible	17903	56	1540	118503	0.5	0.1
627	NoSol	Feasible	30593	56	1540	118503	0.5	0.1	628	NoSol	Feasible	1944	56	1540	237006	0.5	0.2
629	NoSol	Feasible	8837	56	1540	237006	0.5	0.2	630	NoSol	Feasible	13136	56	1540	237006	0.5	0.2
631	21025	Optimal	7893	56	1540	355509	0.5	0.3	632	23439	Optimal	12426	56	1540	355509	0.5	0.3

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
633	20827	Optimal	16483	56	1540	355509	0.5	0.3	634	22175	Optimal	18234	56	1540	474012	0.5	0.4
635	25993	Optimal	19516	56	1540	474012	0.5	0.4	636	26290	Optimal	30522	56	1540	474012	0.5	0.4
637	NoSol	Feasible	6806	56	1848	169924	0.6	0.1	638	NoSol	Feasible	15195	56	1848	169924	0.6	0.1
639	NoSol	Feasible	22937	56	1848	169924	0.6	0.1	640	NoSol	Feasible	10181	56	1848	339848	0.6	0.2
641	NoSol	Feasible	16210	56	1848	339848	0.6	0.2	642	NoSol	Feasible	19579	56	1848	339848	0.6	0.2
643	NoSol	Feasible	1045	56	1848	511619	0.6	0.3	644	NoSol	Feasible	18283	56	1848	511619	0.6	0.3
645	NoSol	Feasible	28734	56	1848	511619	0.6	0.3	646	21954	Optimal	18150	56	1848	681543	0.6	0.4
647	26932	Optimal	19368	56	1848	681543	0.6	0.4	648	20096	Optimal	24699	56	1848	681543	0.6	0.4
649	NoSol	Feasible	6758	56	2156	230585	0.7	0.1	650	NoSol	Feasible	14420	56	2156	230585	0.7	0.1
651	NoSol	Feasible	16184	56	2156	230585	0.7	0.1	652	NoSol	Feasible	1596	56	2156	463325	0.7	0.2
653	NoSol	Feasible	4732	56	2156	463325	0.7	0.2	654	NoSol	Feasible	13622	56	2156	463325	0.7	0.2
655	NoSol	Feasible	4544	56	2156	696065	0.7	0.3	656	NoSol	Feasible	11806	56	2156	696065	0.7	0.3
657	NoSol	Feasible	26304	56	2156	696065	0.7	0.3	658	19780	Optimal	1351	56	2156	928805	0.7	0.4
659	30129	Optimal	8484	56	2156	928805	0.7	0.4	660	25610	Optimal	15862	56	2156	928805	0.7	0.4
661	NoSol	Feasible	4282	56	2464	302949	0.8	0.1	662	NoSol	Feasible	14430	56	2464	302949	0.8	0.1
663	NoSol	Feasible	16420	56	2464	302949	0.8	0.1	664	NoSol	Feasible	1806	56	2464	605898	0.8	0.2
665	NoSol	Feasible	4251	56	2464	605898	0.8	0.2	666	NoSol	Feasible	22760	56	2464	605898	0.8	0.2
667	NoSol	Feasible	4912	56	2464	908847	0.8	0.3	668	NoSol	Feasible	15992	56	2464	908847	0.8	0.3
669	NoSol	Feasible	25663	56	2464	908847	0.8	0.3	670	16142	Optimal	4773	56	2464	1211796	0.8	0.4
671	29475	Optimal	7636	56	2464	1211796	0.8	0.4	672	23379	Optimal	8843	56	2464	1211796	0.8	0.4
673	NoSol	Feasible	21448	65	832	34071	0.2	0.1	674	NoSol	Feasible	22588	65	832	34071	0.2	0.1
675	NoSol	Feasible	32339	65	832	34071	0.2	0.1	676	35295	Optimal	3418	65	832	68973	0.2	0.2
677	39047	Optimal	7217	65	832	68973	0.2	0.2	678	32900	Optimal	16059	65	832	68973	0.2	0.2
679	24369	Optimal	3395	65	832	103044	0.2	0.3	680	26485	Optimal	16772	65	832	103044	0.2	0.3
681	29781	Optimal	16920	65	832	103044	0.2	0.3	682	33628	Optimal	3251	65	832	137946	0.2	0.4
683	29090	Optimal	18861	65	832	137946	0.2	0.4	684	28024	Optimal	19971	65	832	137946	0.2	0.4
685	NoSol	Feasible	247	65	1248	77314	0.3	0.1	686	NoSol	Feasible	4574	65	1248	77314	0.3	0.1
687	NoSol	Feasible	28381	65	1248	77314	0.3	0.1	688	NoSol	Feasible	10011	65	1248	154628	0.3	0.2
689	NoSol	Feasible	15109	65	1248	154628	0.3	0.2	690	NoSol	Feasible	17831	65	1248	154628	0.3	0.2
691	37208	Optimal	14875	65	1248	233189	0.3	0.3	692	29568	Optimal	22837	65	1248	233189	0.3	0.3
693	33799	Optimal	27152	65	1248	233189	0.3	0.3	694	27972	Optimal	14062	65	1248	310503	0.3	0.4
695	33089	Optimal	18863	65	1248	310503	0.3	0.4	696	35851	Optimal	24640	65	1248	310503	0.3	0.4
697	NoSol	Feasible	2691	65	1664	138029	0.4	0.1	698	NoSol	Feasible	17098	65	1664	138029	0.4	0.1
699	NoSol	Feasible	31900	65	1664	138029	0.4	0.1	700	NoSol	Feasible	26	65	1664	276058	0.4	0.2
701	NoSol	Feasible	21774	65	1664	276058	0.4	0.2	702	NoSol	Feasible	28258	65	1664	276058	0.4	0.2
703	23918	Optimal	8930	65	1664	414087	0.4	0.3	704	30664	Optimal	24570	65	1664	414087	0.4	0.3
705	24291	Optimal	27030	65	1664	414087	0.4	0.3	706	34483	Optimal	2908	65	1664	552116	0.4	0.4
707	37600	Optimal	19919	65	1664	552116	0.4	0.4	708	36271	Optimal	22547	65	1664	552116	0.4	0.4
709	NoSol	Feasible	21302	65	2080	216216	0.5	0.1	710	NoSol	Feasible	31084	65	2080	216216	0.5	0.1
711	NoSol	Feasible	31931	65	2080	216216	0.5	0.1	712	NoSol	Feasible	19039	65	2080	432432	0.5	0.2
713	NoSol	Feasible	23180	65	2080	432432	0.5	0.2	714	NoSol	Feasible	32471	65	2080	432432	0.5	0.2
715	26579	Feasible	19609	65	2080	648648	0.5	0.3	716	NoSol	Feasible	28031	65	2080	648648	0.5	0.3
717	NoSol	Feasible	30632	65	2080	648648	0.5	0.3	718	37712	Optimal	7612	65	2080	864864	0.5	0.4
719	29883	Optimal	7885	65	2080	864864	0.5	0.4	720	25538	Optimal	18791	65	2080	864864	0.5	0.4
721	NoSol	Feasible	12521	65	2496	309380	0.6	0.1	722	NoSol	Feasible	16153	65	2496	309380	0.6	0.1
723	NoSol	Feasible	28058	65	2496	309380	0.6	0.1	724	NoSol	Feasible	1747	65	2496	621255	0.6	0.2
725	NoSol	Feasible	7343	65	2496	621255	0.6	0.2	726	NoSol	Feasible	22711	65	2496	621255	0.6	0.2
727	NoSol	Feasible	10001	65	2496	933130	0.6	0.3	728	NoSol	Feasible	12816	65	2496	933130	0.6	0.3
729	NoSol	Feasible	29470	65	2496	933130	0.6	0.3	730	33617	Optimal	639	65	2496	1245005	0.6	0.4
731	29221	Optimal	8215	65	2496	1245005	0.6	0.4	732	40523	Optimal	19966	65	2496	1245005	0.6	0.4
733	NoSol	Feasible	2368	65	2912	422095	0.7	0.1	734	NoSol	Feasible	23093	65	2912	422095	0.7	0.1
735	NoSol	Feasible	27966	65	2912	422095	0.7	0.1	736	NoSol	Feasible	4596	65	2912	847101	0.7	0.2
737	NoSol	Feasible	17774	65	2912	847101	0.7	0.2	738	NoSol	Feasible	30971	65	2912	847101	0.7	0.2
739	NoSol	Feasible	2801	65	2912	1269196	0.7	0.3	740	NoSol	Feasible	19228	65	2912	1269196	0.7	0.3
741	NoSol	Feasible	26652	65	2912	1269196	0.7	0.3	742	24732	Optimal	1346	65	2912	1694202	0.7	0.4
743	31647	Optimal	9105	65	2912	1694202	0.7	0.4	744	23682	Optimal	14159	65	2912	1694202	0.7	0.4
745	NoSol	Feasible	12492	65	3328	552282	0.8	0.1	746	NoSol	Feasible	14112	65	3328	552282	0.8	0.1
747	NoSol	Feasible	24792	65	3328	552282	0.8	0.1	748	NoSol	Feasible	22766	65	3328	1104564	0.8	0.2
749	NoSol	Feasible	26077	65	3328	1104564	0.8	0.2	750	NoSol	Feasible	27057	65	3328	1104564	0.8	0.2

Table A.1. Test instance parameters (cont.).

Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω	Ins.	Best	Status	Seed	|V(N)|	|A(N)|	|E(C)|	Γ	Ω
751	NoSol	Feasible	11209	65	3328	1660173	0.8	0.3	752	NoSol	Feasible	19999	65	3328	1660173	0.8	0.3
753	NoSol	Feasible	22341	65	3328	1660173	0.8	0.3	754	28535	Feasible	3391	65	3328	2212455	0.8	0.4
755	33402	Feasible	3470	65	3328	2212455	0.8	0.4	756	34707	Feasible	32133	65	3328	2212455	0.8	0.4
757	NoSol	Feasible	221	71	994	48657	0.2	0.1	758	NoSol	Feasible	6579	71	994	48657	0.2	0.1
759	NoSol	Feasible	23638	71	994	48657	0.2	0.1	760	32705	Optimal	23591	71	994	98307	0.2	0.2
761	26052	Optimal	29789	71	994	98307	0.2	0.2	762	25915	Optimal	32232	71	994	98307	0.2	0.2
763	26798	Optimal	7070	71	994	147957	0.2	0.3	764	21406	Optimal	20860	71	994	147957	0.2	0.3
765	36729	Optimal	26138	71	994	147957	0.2	0.3	766	25792	Optimal	19475	71	994	196614	0.2	0.4
767	33701	Optimal	20204	71	994	196614	0.2	0.4	768	36685	Optimal	23749	71	994	196614	0.2	0.4
769	NoSol	Feasible	10333	71	1491	110260	0.3	0.1	770	NoSol	Feasible	17725	71	1491	110260	0.3	0.1
771	NoSol	Feasible	28391	71	1491	110260	0.3	0.1	772	NoSol	Feasible	947	71	1491	222010	0.3	0.2
773	NoSol	Feasible	2969	71	1491	222010	0.3	0.2	774	NoSol	Feasible	9428	71	1491	222010	0.3	0.2
775	33302	Optimal	6591	71	1491	332270	0.3	0.3	776	37593	Optimal	16323	71	1491	332270	0.3	0.3
777	31034	Optimal	25375	71	1491	332270	0.3	0.3	778	44324	Optimal	12571	71	1491	444020	0.3	0.4
779	38561	Optimal	14844	71	1491	444020	0.3	0.4	780	40161	Optimal	21029	71	1491	444020	0.3	0.4
781	NoSol	Feasible	23176	71	1988	196713	0.4	0.1	782	NoSol	Feasible	25978	71	1988	196713	0.4	0.1
783	NoSol	Feasible	27671	71	1988	196713	0.4	0.1	784	NoSol	Feasible	8025	71	1988	393426	0.4	0.2
785	NoSol	Feasible	12667	71	1988	393426	0.4	0.2	786	NoSol	Feasible	22901	71	1988	393426	0.4	0.2
787	31113	Optimal	19241	71	1988	592126	0.4	0.3	788	40357	Optimal	19562	71	1988	592126	0.4	0.3
789	28306	Optimal	30945	71	1988	592126	0.4	0.3	790	38115	Optimal	5631	71	1988	788839	0.4	0.4
791	28566	Optimal	26602	71	1988	788839	0.4	0.4	792	34622	Optimal	28727	71	1988	788839	0.4	0.4
793	NoSol	Feasible	9135	71	2485	308016	0.5	0.1	794	NoSol	Feasible	31497	71	2485	308016	0.5	0.1
795	NoSol	Feasible	32516	71	2485	308016	0.5	0.1	796	NoSol	Feasible	5049	71	2485	616032	0.5	0.2
797	NoSol	Feasible	10726	71	2485	616032	0.5	0.2	798	NoSol	Feasible	17484	71	2485	616032	0.5	0.2
799	NoSol	Feasible	4327	71	2485	924048	0.5	0.3	800	NoSol	Feasible	12001	71	2485	924048	0.5	0.3
801	NoSol	Feasible	26863	71	2485	924048	0.5	0.3	802	29694	Optimal	1024	71	2485	1234548	0.5	0.4
803	25143	Optimal	2093	71	2485	1234548	0.5	0.4	804	38372	Optimal	26476	71	2485	1234548	0.5	0.4
805	NoSol	Feasible	7629	71	2982	444169	0.6	0.1	806	NoSol	Feasible	9942	71	2982	444169	0.6	0.1
807	NoSol	Feasible	32504	71	2982	444169	0.6	0.1	808	NoSol	Feasible	2026	71	2982	888338	0.6	0.2
809	NoSol	Feasible	8337	71	2982	888338	0.6	0.2	810	NoSol	Feasible	8788	71	2982	888338	0.6	0.2
811	NoSol	Feasible	790	71	2982	1332507	0.6	0.3	812	NoSol	Feasible	13373	71	2982	1332507	0.6	0.3
813	NoSol	Feasible	22018	71	2982	1332507	0.6	0.3	814	40486	Optimal	11800	71	2982	1776676	0.6	0.4
815	27711	Optimal	15422	71	2982	1776676	0.6	0.4	816	39745	Optimal	25076	71	2982	1776676	0.6	0.4
817	NoSol	Feasible	16193	71	3479	601694	0.7	0.1	818	NoSol	Feasible	17916	71	3479	601694	0.7	0.1
819	NoSol	Feasible	22108	71	3479	601694	0.7	0.1	820	NoSol	Feasible	14022	71	3479	1206866	0.7	0.2
821	NoSol	Feasible	16400	71	3479	1206866	0.7	0.2	822	NoSol	Feasible	27623	71	3479	1206866	0.7	0.2
823	NoSol	Feasible	4715	71	3479	1812038	0.7	0.3	824	NoSol	Feasible	12156	71	3479	1812038	0.7	0.3
825	NoSol	Feasible	26728	71	3479	1812038	0.7	0.3	826	34783	Feasible	7228	71	3479	2417210	0.7	0.4
827	35724	Feasible	7567	71	3479	2417210	0.7	0.4	828	42560	Feasible	29671	71	3479	2417210	0.7	0.4
829	NoSol	Feasible	4448	71	3976	787050	0.8	0.1	830	NoSol	Feasible	10109	71	3976	787050	0.8	0.1
831	NoSol	Feasible	20265	71	3976	787050	0.8	0.1	832	NoSol	Feasible	5634	71	3976	1578075	0.8	0.2
833	NoSol	Feasible	5691	71	3976	1578075	0.8	0.2	834	NoSol	Feasible	31313	71	3976	1578075	0.8	0.2
835	NoSol	Feasible	2567	71	3976	2369100	0.8	0.3	836	NoSol	Feasible	8293	71	3976	2369100	0.8	0.3
837	NoSol	Feasible	15143	71	3976	2369100	0.8	0.3	838	NoSol	Feasible	3886	71	3976	3160125	0.8	0.4
839	33736	Feasible	9372	71	3976	3160125	0.8	0.4	840	NoSol	Feasible	25672	71	3976	3160125	0.8	0.4

APPENDIX B: DETAILED RESULTS

Detailed experimental results obtained with BBA, BBA w/o inf., Diving, Weak - single, Weak-multi, and BBSS are reported in Table B.1. Row numbers indicate the instance id which is between 1 and 420 for random test instances and between 421 and 840 for realistic test instances. "Best UB" is the best solution's objective value reported by respective algorithm, showing "NA" if no solution is reported. Run times in seconds are reported in "CPU" column.

Table B.1. Detailed experimental results.

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
1	719	0.0	719	3.0	719	53.0	719	2.1	719	1.0	719	0.7
2	1016	0.0	1016	0.8	1016	172.0	1016	1.3	1016	0.3	1016	0.1
3	917	0.0	917	1.4	917	965.0	917	4.6	917	1.2	917	0.5
4	1515	0.0	1515	0.1	1515	0.0	1515	0.0	1515	0.0	1515	0.0
5	1335	0.0	1335	0.4	1335	0.0	1335	0.0	1335	0.0	1335	0.0
6	1065	0.0	1065	0.2	1065	0.0	1065	0.0	1065	0.0	1065	0.0
7	1365	0.0	1365	0.1	1365	0.0	1365	0.0	1365	0.0	1365	0.0
8	1620	0.0	1620	0.1	1620	0.0	1620	0.0	1620	0.0	1620	0.0
9	1665	0.0	1665	0.0	1665	0.0	1665	0.0	1665	0.0	1665	0.0
10	1515	0.0	1515	0.0	1515	0.0	1515	0.0	1515	0.0	1515	0.0
11	1350	0.0	1350	0.0	1350	0.0	1350	0.0	1350	0.0	1350	0.0
12	1215	0.0	1215	0.0	1215	0.0	1215	0.0	1215	0.0	1215	0.0
13	345	0.0	345	0.9	NA	1800.0	345	4.5	345	2.2	345	138.3
14	431	0.0	431	0.7	942	1802.0	431	3.4	431	2.0	431	45.5
15	517	1.0	517	4.5	744	1804.0	517	11.2	517	10.8	517	28.5
16	1260	1.0	1260	30.7	1260	2.0	1260	26.1	1260	7.0	1260	2.0
17	1485	0.0	1485	9.8	1485	4.0	1485	6.2	1485	2.8	1485	0.4
18	1126	1.0	1126	13.9	1126	4.0	1126	26.5	1126	10.0	1126	1.5
19	1170	0.0	1170	0.2	1170	0.0	1170	0.0	1170	0.1	1170	0.0
20	1950	0.0	1950	1.1	1950	0.0	1950	0.1	1950	0.1	1950	0.1
21	1390	0.0	1390	1.0	1390	0.0	1390	0.1	1390	0.1	1390	0.0
22	1215	0.0	1215	0.1	1215	0.0	1215	0.0	1215	0.1	1215	0.0
23	1170	0.0	1170	0.1	1170	0.0	1170	0.0	1170	0.1	1170	0.0
24	1395	0.0	1395	0.1	1395	0.0	1395	0.1	1395	0.1	1395	0.0
25	347	7.0	347	7.8	NA	1800.0	347	104.7	347	11.7	354	1800.0
26	281	5.0	281	4.8	NA	1800.0	281	42.0	281	28.3	294	1800.0
27	360	6.0	360	5.4	NA	1800.0	360	75.1	360	19.0	543	1800.0
28	1320	45.0	NA	1803.0	1320	39.0	1320	255.1	1320	50.5	1320	74.1
29	1149	28.0	1149	446.9	1149	76.0	1149	202.6	1149	33.8	1149	61.6
30	1575	118.0	NA	1800.0	1575	193.0	1575	334.7	1575	110.8	1575	186.0
31	1095	0.0	1095	1.5	1095	0.0	1095	43.8	1095	6.5	1095	0.9
32	1335	0.0	1335	5.1	1335	0.0	1335	43.1	1335	7.6	1335	0.5
33	1020	0.0	1020	1.8	1020	0.0	1020	34.9	1020	13.9	1020	0.9
34	1560	0.0	1560	0.6	1560	0.0	1560	36.1	1560	1.8	1560	0.2
35	1080	0.0	1080	0.4	1080	0.0	1080	21.8	1080	1.9	1080	0.2
36	1665	0.0	1665	0.4	1665	0.0	1665	25.6	1665	1.8	1665	0.2

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
37	239	0.0	239	0.7	NA	1800.0	239	5.7	239	2.2	407	1800.0
38	306	18.0	306	12.8	NA	1800.0	306	145.4	306	47.9	509	1800.0
39	296	41.0	296	28.0	NA	1800.0	296	148.4	296	47.1	371	1800.0
40	1157	432.0	NA	1800.0	1176	1800.0	1157	1800.0	1157	1800.0	1157	1800.0
41	1005	1185.0	NA	1800.0	NA	1805.0	NA	1800.0	1005	1556.9	1005	1423.3
42	1095	1067.0	NA	1800.0	NA	1800.0	NA	1800.0	1095	1800.0	1095	1800.0
43	1575	2.0	1575	22.7	1575	3.0	1575	85.8	1575	25.2	1575	12.0
44	1044	5.0	1044	26.0	1044	0.0	1044	191.5	1044	24.9	1044	13.3
45	1755	5.0	1755	41.1	1755	1.0	1755	144.6	1755	21.0	1755	17.6
46	1365	0.0	1365	1.9	1365	0.0	1365	71.6	1365	23.5	1365	0.6
47	1035	0.0	1035	2.0	1035	0.0	1035	64.4	1035	19.6	1035	0.5
48	1680	0.0	1680	2.2	1680	0.0	1680	66.7	1680	15.2	1680	0.7
49	197	24.0	197	14.4	NA	1800.0	197	135.2	197	38.5	357	1800.0
50	236	12.0	236	9.2	NA	1800.0	236	117.3	236	37.8	315	1800.0
51	258	16.0	258	9.3	NA	1800.0	258	159.3	258	17.0	361	1800.0
52	491	1077.0	NA	1800.0	1182	1800.0	507	1800.0	517	1801.2	NA	1800.0
53	584	658.0	NA	1800.0	NA	1800.0	586	1800.0	586	1800.0	721	1800.0
54	NA	1800.0	NA	1800.0	NA	1801.0	NA	1800.0	775	1800.0	735	1800.0
55	1245	38.0	1245	411.4	1245	12.0	1245	514.3	1245	123.6	1245	118.7
56	1215	54.0	1215	467.2	1215	26.0	1215	738.4	1215	380.0	1215	109.5
57	1275	32.0	1275	299.3	1275	27.0	1275	351.7	1275	138.5	1275	78.6
58	1605	1.0	1605	9.9	1605	1.0	1605	97.5	1605	32.0	1605	5.9
59	1365	1.0	1365	4.1	1365	0.0	1365	126.7	1365	69.0	1365	3.4
60	1290	1.0	1290	6.2	1290	0.0	1290	187.2	1290	25.6	1290	4.4
61	167	13.0	167	8.1	NA	1800.0	167	85.4	167	42.0	272	1800.0
62	211	17.0	211	10.4	NA	1800.0	211	91.1	211	37.1	359	1800.0
63	159	15.0	159	10.8	NA	1800.0	159	67.9	159	47.8	351	1800.0
64	447	1134.0	NA	1800.0	NA	1802.0	487	1800.0	509	1800.3	785	1800.0
65	NA	1800.0	NA	1800.0	1048	1800.0	819	1800.0	658	1800.0	1036	1800.0
66	437	539.0	437	1018.7	NA	1801.0	465	1800.0	471	1800.0	1303	1800.0
67	1560	256.0	NA	1800.0	1560	121.0	NA	1800.0	1560	1202.6	1560	737.2
68	1425	224.0	NA	1800.0	1425	61.0	NA	1800.0	1425	880.3	1425	687.7
69	1680	94.0	1680	988.2	1680	47.0	NA	1800.0	1680	1307.9	1680	640.8
70	1455	10.0	1455	74.5	1455	1.0	1455	433.9	1455	63.0	1455	26.7
71	1080	6.0	1080	37.6	1080	1.0	1080	379.3	1080	76.3	1080	22.8
72	1425	7.0	1425	31.3	1425	1.0	1425	265.2	1425	68.0	1425	29.7
73	224	25.0	224	14.8	NA	1800.0	224	234.1	224	73.0	377	1800.0
74	164	10.0	164	6.9	NA	1800.0	164	86.3	164	33.3	250	1800.0
75	156	66.0	156	36.8	NA	1800.0	156	359.5	156	114.2	321	1800.0
76	NA	1800.0	NA	1800.5	NA	1800.0	515	1800.0	599	1801.1	672	1800.0
77	328	447.0	328	570.2	NA	1800.0	328	1800.0	328	1800.0	941	1800.0
78	NA	1800.0	NA	1800.0	NA	1800.0	403	1800.0	408	1800.0	1233	1800.0
79	1605	839.0	NA	1800.0	1605	343.0	NA	1800.0	NA	1800.1	NA	1800.0
80	1560	379.0	NA	1800.0	1560	322.0	NA	1800.0	NA	1800.0	1560	1800.0
81	1185	1367.0	NA	1800.0	1185	908.0	NA	1800.0	NA	1800.0	NA	1800.0
82	1215	11.0	1215	64.1	1215	1.0	1215	851.9	1215	181.1	1215	69.1
83	1305	24.0	1305	127.1	1305	2.0	1305	839.3	1305	218.3	1305	78.2
84	1695	13.0	1695	77.5	1695	7.0	1695	538.5	1695	148.9	1695	85.8
85	976	18.0	976	1224.4	1439	1800.0	976	778.9	976	196.3	976	203.3
86	1038	47.0	NA	1800.0	1565	1800.0	1038	827.2	1038	76.7	1038	52.4
87	1247	85.0	NA	1800.0	NA	1800.0	1256	1800.0	1247	1800.0	1247	322.8
88	2130	0.0	2130	10.2	2130	0.0	2130	0.1	2130	0.1	2130	0.1
89	2325	0.0	2325	5.0	2325	1.0	2325	0.1	2325	0.0	2325	0.3
90	1545	0.0	1545	2.3	1545	2.0	1545	0.6	1545	0.3	1545	0.3
91	1875	0.0	1875	0.5	1875	0.0	1875	0.0	1875	0.1	1875	0.0
92	1980	0.0	1980	0.2	1980	0.0	1980	0.1	1980	0.1	1980	0.0
93	2175	0.0	2175	0.3	2175	0.0	2175	0.1	2175	0.1	2175	0.0
94	1755	0.0	1755	0.1	1755	0.0	1755	0.1	1755	0.1	1755	0.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
95	1770	0.0	1770	0.1	1770	0.0	1770	0.1	1770	0.1	1770	0.0
96	1350	0.0	1350	0.1	1350	0.0	1350	0.1	1350	0.1	1350	0.0
97	473	1286.0	NA	1800.0	NA	1800.0	482	1800.0	473	1800.0	540	1800.0
98	NA	1800.0	NA	1800.0	NA	1800.0	705	1800.0	624	1800.0	1042	1800.0
99	NA	1800.0	NA	1800.0	2226	1800.0	701	1800.0	752	1800.0	1018	1800.0
100	1489	49.0	1489	1230.6	1489	104.0	1489	187.7	1489	30.7	1489	125.1
101	1935	135.0	NA	1800.0	1935	72.0	1935	196.8	1935	25.3	1935	118.7
102	2010	100.0	NA	1800.0	2010	362.0	2010	163.4	2010	37.6	2010	85.5
103	1755	1.0	1755	8.6	1755	0.0	1755	54.8	1755	7.1	1755	1.5
104	2595	0.0	2595	3.2	2595	0.0	2595	31.0	2595	2.4	2595	0.9
105	2070	0.0	2070	2.4	2070	0.0	2070	126.7	2070	21.4	2070	1.5
106	2115	0.0	2115	0.5	2115	0.0	2115	2.1	2115	1.4	2115	0.2
107	2190	0.0	2190	1.4	2190	0.0	2190	1.3	2190	1.4	2190	0.3
108	1680	0.0	1680	0.9	1680	0.0	1680	58.7	1680	2.6	1680	0.2
109	NA	1800.0	NA	1800.0	NA	1800.0	482	1800.0	491	1800.0	767	1800.0
110	NA	1800.0	NA	1800.0	NA	1800.0	577	1800.0	589	1800.0	840	1800.0
111	401	1807.0	401	1800.0	NA	1800.0	505	1800.0	410	1800.0	878	1800.0
112	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	1527	1800.0
113	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
114	NA	1800.0	NA	1800.0	1890	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
115	2295	4.0	2295	115.8	2295	7.0	2295	383.2	2295	54.0	2295	76.1
116	1770	26.0	1770	473.0	1770	3.0	1770	301.9	1770	163.4	1770	64.6
117	2025	14.0	2025	195.0	2025	8.0	2025	550.0	2025	129.8	2025	90.7
118	2130	0.0	2130	5.2	2130	0.0	2130	345.6	2130	27.9	2130	3.1
119	1710	1.0	1710	5.6	1710	0.0	1710	331.9	1710	42.3	1710	2.9
120	2415	1.0	2415	5.8	2415	0.0	2415	216.8	2415	37.1	2415	1.9
121	NA	1800.0	NA	1800.0	NA	1800.0	406	1800.0	373	1801.0	696	1800.0
122	NA	1800.0	NA	1800.0	NA	1800.0	409	1800.0	409	1800.0	956	1800.0
123	NA	1800.0	NA	1800.0	NA	1800.0	390	1800.0	390	1800.1	987	1800.0
124	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
125	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
126	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
127	2415	44.0	2415	1197.8	2415	63.0	2415	946.7	2415	245.5	2415	1053.4
128	1860	178.0	NA	1800.0	1860	40.0	1860	1781.6	1860	1090.9	1860	848.4
129	1965	169.0	NA	1800.0	1965	72.0	NA	1800.0	1965	1081.3	1965	920.4
130	1830	2.0	1830	14.1	1830	0.0	1830	436.8	1830	51.0	1830	17.3
131	2430	3.0	2430	19.6	2430	0.0	2430	375.9	2430	71.7	2430	24.9
132	1860	3.0	1860	13.9	1860	0.0	1860	373.6	1860	75.1	1860	27.4
133	NA	1800.0	NA	1800.0	NA	1800.0	330	1800.0	308	1800.3	743	1800.0
134	NA	1800.0	NA	1800.0	NA	1800.0	244	1800.0	251	1800.0	1039	1800.0
135	NA	1800.0	NA	1800.0	NA	1800.0	266	1800.0	266	1153.5	732	1800.0
136	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
137	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
138	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
139	1875	1094.0	NA	1800.0	1875	136.0	NA	1800.0	NA	1800.0	NA	1800.0
140	NA	1800.0	NA	1800.0	2115	308.0	NA	1800.0	NA	1800.1	NA	1800.0
141	NA	1800.0	NA	1800.0	1740	236.0	NA	1800.0	NA	1800.1	NA	1800.0
142	1875	18.0	1875	225.8	1875	9.0	1875	1643.9	1875	192.3	1875	199.2
143	1785	31.0	1785	214.9	1785	3.0	1785	1704.5	1785	1302.9	1785	216.1
144	2160	57.0	2160	568.9	2160	2.0	NA	1800.0	2160	252.8	2160	161.0
145	NA	1800.0	NA	1800.0	NA	1800.0	289	1800.0	289	1800.2	774	1800.0
146	NA	1800.0	NA	1800.0	NA	1800.0	354	1800.0	332	1800.0	613	1800.0
147	NA	1800.0	NA	1801.0	NA	1800.0	243	1800.0	242	1800.1	447	1800.0
148	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
149	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
150	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
151	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
152	NA	1800.0	NA	1800.1	2490	851.0	NA	1800.0	NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
153	NA	1800.0	NA	1800.0	2580	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
154	1470	32.0	1470	193.6	1470	38.0	NA	1800.0	1470	988.5	1470	1177.2
155	1965	88.0	1965	614.5	1965	7.0	NA	1800.0	1965	1241.1	1965	1013.8
156	2220	183.0	2220	1631.9	2220	19.0	NA	1800.0	2220	1161.5	2220	1175.1
157	NA	1800.0	NA	1800.0	NA	1800.0	273	1800.0	239	1800.4	701	1800.0
158	NA	1800.0	NA	1801.4	NA	1800.0	238	1800.0	238	1800.1	367	1800.0
159	NA	1800.0	NA	1800.0	NA	1800.0	317	1800.0	309	1800.1	668	1800.0
160	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
161	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
162	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
163	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
164	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
165	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
166	1875	203.0	NA	1800.0	1875	72.0	NA	1800.0	NA	1800.1	1875	1800.0
167	2415	266.0	NA	1800.0	2415	107.0	NA	1800.0	NA	1800.0	2415	1800.0
168	1830	232.0	NA	1800.0	1830	125.0	NA	1800.0	NA	1800.1	NA	1800.0
169	NA	1800.0	NA	1800.0	NA	1800.0	1403	1800.0	1260	1800.0	NA	1800.0
170	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	1749	1800.0	NA	1800.0
171	NA	1800.0	NA	1800.0	NA	1800.0	1543	1800.0	1543	1800.0	NA	1800.0
172	2445	7.0	2445	923.0	2445	61.0	2445	377.8	2445	17.3	2445	21.2
173	2775	1.0	2775	75.1	2775	29.0	2775	0.5	2775	0.3	2775	7.4
174	2361	3.0	2361	122.9	2361	15.0	2361	104.9	2361	19.5	2361	7.8
175	3060	0.0	3060	0.8	3060	0.0	3060	1.9	3060	0.7	3060	0.6
176	3870	0.0	3870	8.7	3870	0.0	3870	2.4	3870	1.0	3870	0.2
177	2235	0.0	2235	2.0	2235	0.0	2235	1.6	2235	0.7	2235	0.7
178	2385	0.0	2385	0.3	2385	0.0	2385	0.5	2385	0.3	2385	0.1
179	2220	0.0	2220	0.4	2220	0.0	2220	0.6	2220	0.3	2220	0.1
180	1890	0.0	1890	0.4	1890	0.0	1890	0.5	1890	0.2	1890	0.0
181	NA	1800.0	NA	1800.0	NA	1800.0	1151	1800.0	993	1800.0	NA	1800.0
182	NA	1800.0	NA	1800.0	NA	1800.0	1023	1800.0	1056	1800.1	NA	1800.0
183	NA	1800.0	NA	1800.0	NA	1800.0	698	1800.0	710	1800.0	2484	1800.0
184	2820	586.0	NA	1800.0	2820	1400.0	2820	1800.0	2820	624.0	NA	1800.0
185	3210	1722.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	3210	1800.0
186	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	2731	1800.1	NA	1800.0
187	2415	14.0	2415	660.4	2415	2.0	2415	501.6	2415	44.9	2415	72.8
188	2265	9.0	2265	126.4	2265	1.0	2265	439.1	2265	49.9	2265	28.7
189	2985	7.0	2985	151.7	2985	1.0	2985	674.7	2985	72.4	2985	54.9
190	2325	1.0	2325	15.1	2325	0.0	2325	714.8	2325	12.1	2325	3.4
191	2415	0.0	2415	2.1	2415	0.0	2415	1502.0	2415	84.8	2415	3.4
192	2940	0.0	2940	2.1	2940	0.0	2940	698.2	2940	10.8	2940	2.3
193	NA	1800.0	NA	1800.0	NA	1800.0	681	1800.0	500	1800.0	873	1800.0
194	NA	1800.0	NA	1800.0	NA	1800.0	1115	1800.0	769	1800.1	1562	1800.0
195	NA	1800.0	NA	1800.0	NA	1800.0	911	1800.0	721	1800.1	NA	1800.0
196	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
197	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
198	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
199	2895	193.0	NA	1800.0	2895	26.0	NA	1800.0	2895	711.0	NA	1800.0
200	2190	142.0	NA	1800.0	2190	121.0	NA	1800.0	2190	593.9	2190	1158.1
201	2595	224.0	NA	1800.0	2595	150.0	NA	1800.0	2595	669.4	2595	1704.6
202	2415	2.0	2415	16.2	2415	0.0	2415	1434.5	2415	135.8	2415	56.9
203	2715	10.0	2715	85.7	2715	2.0	NA	1800.0	2715	120.0	2715	37.2
204	3750	5.0	3750	34.1	3750	2.0	3750	1800.0	3750	140.0	3750	56.9
205	NA	1800.0	NA	1800.0	NA	1800.0	575	1800.0	487	1800.1	1751	1800.0
206	NA	1800.0	NA	1800.0	NA	1800.0	581	1800.0	483	1800.0	NA	1800.0
207	NA	1800.0	NA	1800.0	NA	1800.0	792	1800.0	755	1800.1	1393	1800.0
208	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
209	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
210	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
211	NA	1800.0	NA	1800.0	2430	563.0	NA	1800.0	NA	1800.0	NA	1800.0
212	NA	1800.0	NA	1800.0	2145	273.0	NA	1800.0	NA	1800.1	NA	1800.0
213	NA	1800.0	NA	1800.0	2205	1136.0	NA	1800.0	NA	1800.2	NA	1800.0
214	2640	60.0	2640	758.6	2640	3.0	NA	1800.0	2640	1800.0	2640	534.6
215	2655	55.0	2655	565.0	2655	9.0	NA	1800.0	2655	1800.1	2655	691.5
216	2580	36.0	2580	215.1	2580	9.0	NA	1800.0	2580	1060.9	2580	620.9
217	NA	1800.0	NA	1800.0	NA	1800.0	573	1800.0	403	1800.1	1158	1800.0
218	NA	1800.0	NA	1800.0	NA	1800.0	549	1800.0	623	1800.1	1271	1800.0
219	NA	1800.0	NA	1800.0	NA	1800.0	478	1800.0	417	1800.1	NA	1800.0
220	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
221	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
222	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
223	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
224	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
225	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
226	3435	153.0	3435	1614.2	3435	36.0	NA	1800.0	3435	1800.1	NA	1800.0
227	2280	381.0	NA	1800.0	2280	62.0	NA	1800.1	2280	1800.1	NA	1800.0
228	2745	283.0	NA	1800.0	2745	27.0	NA	1800.0	NA	1800.0	NA	1800.0
229	NA	1800.0	NA	1800.3	NA	1800.0	580	1800.0	407	1800.1	984	1800.0
230	NA	1800.0	NA	1800.0	NA	1800.0	773	1800.1	618	1800.0	NA	1800.0
231	NA	1800.0	NA	1800.0	NA	1800.0	496	1800.1	454	1800.1	2234	1800.0
232	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
233	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
234	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
235	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
236	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
237	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
238	2565	1040.0	NA	1800.0	2565	179.0	NA	1800.0	NA	1800.1	NA	1800.0
239	NA	1800.0	NA	1800.0	2565	252.0	NA	1800.0	NA	1800.1	2565	1800.0
240	2910	1467.0	NA	1800.0	2910	115.0	NA	1800.0	NA	1800.1	NA	1800.0
241	NA	1800.0	NA	1800.0	NA	1800.0	546	1800.1	474	1800.1	1458	1800.0
242	NA	1800.0	NA	1800.0	NA	1800.0	550	1800.3	406	1800.2	1450	1800.0
243	NA	1800.0	NA	1800.0	NA	1800.0	686	1800.1	325	1800.1	774	1800.0
244	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
245	NA	1800.0	NA	1800.2	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
246	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
247	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
248	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
249	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
250	NA	1800.0	NA	1800.0	2520	542.0	NA	1800.1	NA	1800.1	NA	1800.0
251	NA	1800.0	NA	1800.0	2490	1394.0	NA	1800.0	NA	1800.1	NA	1800.0
252	NA	1800.0	NA	1800.0	2805	799.0	NA	1800.0	NA	1800.1	NA	1800.0
253	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
254	NA	1800.0	NA	1800.2	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
255	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
256	2925	37.0	NA	1800.0	2925	144.0	2925	534.6	2925	123.6	2925	394.9
257	3289	95.0	NA	1800.0	3289	309.0	3289	679.0	3289	64.9	3289	396.4
258	3405	138.0	NA	1800.0	3405	44.0	3405	760.0	3405	77.0	3405	244.0
259	2970	1.0	2970	24.3	2970	0.0	2970	644.9	2970	121.4	2970	5.1
260	3435	0.0	3435	4.7	3435	0.0	3435	242.8	3435	6.9	3435	3.4
261	2970	0.0	2970	15.6	2970	0.0	2970	466.7	2970	76.3	2970	3.4
262	3735	0.0	3735	0.9	3735	0.0	3735	1.5	3735	0.9	3735	0.2
263	4005	0.0	4005	0.7	4005	0.0	4005	3.2	4005	2.3	4005	0.4
264	3540	0.0	3540	1.0	3540	0.0	3540	1.7	3540	1.6	3540	0.5
265	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0
266	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
267	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
268	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
269	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
270	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
271	2445	93.0	2445	1381.7	2445	33.0	NA	1800.0	2445	566.7	2445	770.7
272	3360	66.0	3360	1463.5	3360	23.0	NA	1800.0	3360	272.0	3360	614.8
273	3495	90.0	NA	1800.0	3495	19.0	NA	1800.0	3495	362.7	3495	1048.4
274	3540	2.0	3540	14.0	3540	1.0	NA	1800.0	3540	191.1	3540	31.6
275	3285	3.0	3285	10.1	3285	0.0	NA	1800.0	3285	241.1	3285	28.4
276	2580	3.0	2580	14.9	2580	2.0	NA	1800.0	2580	153.1	2580	34.0
277	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
278	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
279	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
280	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
281	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
282	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
283	2805	893.0	NA	1800.0	2805	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
284	2730	992.0	NA	1800.0	2730	1104.0	NA	1800.0	NA	1800.1	NA	1800.0
285	NA	1800.0	NA	1800.0	3135	366.0	NA	1800.0	NA	1800.0	NA	1800.0
286	2595	15.0	2595	116.0	2595	6.0	NA	1800.0	2595	1800.1	2595	548.9
287	2775	40.0	2775	437.2	2775	13.0	NA	1800.2	2775	657.1	2775	522.3
288	3375	41.0	3375	317.5	3375	4.0	NA	1800.0	3375	988.0	3375	591.0
289	NA	1800.0	NA	1800.4	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
290	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
291	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	1119	1800.0
292	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
293	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
294	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
295	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
296	NA	1800.0	NA	1800.0	3270	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
297	NA	1800.0	NA	1800.0	3315	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
298	3315	147.0	3315	1273.2	3315	33.0	NA	1800.0	NA	1800.1	NA	1800.0
299	3360	372.0	NA	1800.0	3360	52.0	NA	1800.0	NA	1800.1	NA	1800.0
300	2805	1184.0	NA	1800.0	2805	34.0	NA	1800.3	NA	1800.1	NA	1800.0
301	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
302	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
303	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
304	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
305	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
306	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.5	NA	1800.0
307	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
308	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
309	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
310	NA	1800.0	NA	1800.0	2940	201.0	NA	1800.1	NA	1800.4	NA	1800.0
311	NA	1800.0	NA	1800.0	3435	421.0	NA	1800.0	NA	1800.2	NA	1800.0
312	NA	1800.0	NA	1800.0	3480	217.0	NA	1800.0	NA	1800.5	NA	1800.0
313	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
314	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0
315	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
316	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
317	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
318	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
319	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.1	NA	1800.0
320	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
321	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
322	NA	1800.0	NA	1800.0	2430	1800.0	NA	1800.0	NA	1800.6	NA	1800.0
323	NA	1800.0	NA	1800.0	2835	764.0	NA	1800.1	NA	1800.2	NA	1800.0
324	NA	1800.0	NA	1800.0	3450	1249.0	NA	1800.2	NA	1800.3	NA	1800.0
325	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
326	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.3	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
327	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.1	782	1800.0
328	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
329	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
330	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
331	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.1	NA	1800.0
332	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
333	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
334	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
335	NA	1800.0	NA	1800.0	3000	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
336	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.1	NA	1800.0
337	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
338	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.2	NA	1800.0	NA	1800.0
339	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
340	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	3255	1800.0	NA	1800.0
341	3135	1747.0	NA	1801.6	3135	213.0	NA	1800.0	3135	1800.0	NA	1800.0
342	3840	808.0	NA	1800.0	3840	859.0	3840	1696.5	3840	265.0	NA	1800.0
343	3585	4.0	3585	106.9	3585	0.0	3585	1800.0	3585	78.1	3585	19.2
344	5970	4.0	5970	54.0	5970	3.0	5970	1800.0	5970	90.2	5970	65.6
345	3390	10.0	3390	98.2	3390	5.0	NA	1800.0	3390	149.0	3390	67.2
346	4020	1.0	4020	14.2	4020	0.0	4020	1800.0	4020	20.9	4020	3.0
347	5805	0.0	5805	5.1	5805	0.0	5805	20.7	5805	9.9	5805	1.6
348	3420	0.0	3420	2.5	3420	0.0	3420	1230.3	3420	14.2	3420	1.8
349	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
350	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
351	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0	NA	1800.0
352	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
353	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
354	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
355	NA	1800.0	NA	1800.0	4545	300.0	NA	1800.0	NA	1800.4	NA	1800.0
356	4365	412.0	NA	1800.0	4365	224.0	NA	1800.0	NA	1800.0	NA	1800.0
357	NA	1800.0	NA	1800.0	3390	351.0	NA	1800.0	NA	1800.0	NA	1800.0
358	4005	12.0	4005	91.2	4005	0.0	NA	1800.0	4005	15.7	4005	231.1
359	3405	16.0	3405	145.5	3405	2.0	NA	1800.1	3405	386.2	3405	249.5
360	4200	15.0	4200	181.2	4200	5.0	NA	1800.0	4200	390.8	4200	181.5
361	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0
362	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
363	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
364	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
365	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
366	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
367	NA	1800.0	NA	1800.0	3090	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
368	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
369	NA	1800.0	NA	1800.0	3825	1607.0	NA	1800.0	NA	1800.1	NA	1800.0
370	3990	394.0	NA	1800.0	3990	33.0	NA	1800.0	NA	1800.1	3990	1800.0
371	4695	391.0	NA	1800.0	4695	15.0	NA	1800.0	NA	1800.1	4695	1800.0
372	4470	508.0	NA	1800.0	4470	50.0	NA	1800.0	NA	1800.1	NA	1800.0
373	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
374	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
375	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.5	NA	1800.0
376	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
377	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
378	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
379	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
380	NA	1800.0	NA	1800.0	4575	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
381	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
382	NA	1800.0	NA	1800.0	3900	341.0	NA	1800.0	NA	1800.4	NA	1800.0
383	NA	1800.0	NA	1800.0	4980	285.0	NA	1800.1	NA	1800.1	NA	1800.0
384	NA	1800.0	NA	1800.0	4350	351.0	NA	1800.1	NA	1800.1	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.		Diving		Weak - single		Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
385	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
386	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
387	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
388	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
389	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
390	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
391	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.3	NA	1800.0
392	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
393	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.5	NA	1800.0
394	NA	1800.0	NA	1800.0	3405	1391.0	NA	1800.1	NA	1800.2	NA	1800.0
395	NA	1800.0	NA	1800.0	5085	1569.0	NA	1800.1	NA	1800.1	NA	1800.0
396	NA	1800.0	NA	1800.0	4470	1037.0	NA	1800.0	NA	1800.1	NA	1800.0
397	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
398	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
399	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.5	NA	1800.0
400	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.1	NA	1800.0
401	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.5	NA	1800.1	NA	1800.0
402	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.1	NA	1800.0
403	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
404	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
405	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.0
406	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.1	NA	1800.0
407	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
408	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.2	NA	1800.0
409	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
410	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
411	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
412	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.1	NA	1800.0
413	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.4	NA	1800.0
414	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.3	NA	1800.0
415	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.2	NA	1800.1	NA	1800.0
416	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.3	NA	1800.0
417	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.0
418	NA	1800.0	NA	1800.0	NA	1800.0	NA	1801.1	NA	1800.1	NA	1800.0
419	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.1	NA	1800.0
420	NA	1800.0	NA	1800.0	NA	1800.0	NA	1800.1	NA	1800.3	NA	1800.0
421	15203	2.1			15203	115.0			15203	15.9	15203	4.0
422	10015	0.7			10015	10.0			10015	3.8	10015	2.1
423	8976	3.5			8976	833.0			8976	24.0	8976	7.6
424	10707	0.0			10707	0.0			10707	0.0	10707	0.0
425	11710	0.0			11710	0.0			11710	0.0	11710	0.1
426	12065	0.0			12065	0.0			12065	0.0	12065	0.0
427	7298	0.0			7298	0.0			7298	0.0	7298	0.0
428	12483	0.0			12483	0.0			12483	0.0	12483	0.0
429	15942	0.0			15942	0.0			15942	0.0	15942	0.0
430	15789	0.0			15789	0.0			15789	0.0	15789	0.0
431	14569	0.0			14569	0.0			14569	0.0	14569	0.0
432	12254	0.0			12254	0.0			12254	0.0	12254	0.0
433	9089	339.1			11426	1800.0			9089	1800.0	10271	1800.0
434	5724	87.5			NA	1800.0			5724	72.2	11178	1800.0
435	7071	50.0			12919	1800.0			7071	79.1	13606	1800.0
436	11467	1.7			11467	26.0			11467	13.4	11467	4.0
437	8896	0.8			8896	1.0			8896	16.4	8896	17.5
438	10486	0.4			10486	2.0			10486	10.3	10486	21.1
439	15683	0.1			15683	0.0			15683	0.9	15683	0.3
440	9621	0.1			9621	0.0			9621	1.7	9621	0.3
441	12553	0.0			12553	0.0			12553	0.7	12553	0.4
442	11300	0.0			11300	0.0			11300	0.1	11300	0.1

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
443	15692	0.0		15692	0.0		15692	0.1	15692	0.1
444	9550	0.0		9550	0.0		9550	0.1	9550	0.0
445	4841	1800.4		9591	1800.0		4841	1747.4	9885	1800.0
446	5553	544.3		NA	1800.0		5553	1180.7	8640	1800.0
447	4887	356.9		NA	1800.0		4887	459.0	10745	1800.0
448	11275	32.2		11275	108.0		11275	1800.0	11275	1018.0
449	18389	48.8		18389	67.0		18389	304.5	18389	799.6
450	9702	25.9		9702	338.0		9702	896.4	9702	1667.8
451	12382	0.6		12382	1.0		12382	15.2	12382	7.6
452	10856	0.6		10856	1.0		10856	20.2	10856	9.7
453	8944	0.6		8944	0.0		8944	18.6	8944	11.6
454	19163	0.2		19163	0.0		19163	2.7	19163	0.6
455	11900	0.1		11900	0.0		11900	2.3	11900	0.4
456	19822	0.2		19822	0.0		19822	2.4	19822	0.5
457	4884	1641.1		NA	1800.0		4884	1481.9	11805	1800.0
458	4910	1514.8		NA	1800.0		4910	454.8	10370	1800.0
459	NA	1800.0		NA	1800.0		4661	676.4	10287	1800.0
460	7239	80.6		7239	1772.0		7239	1800.0	NA	1800.0
461	19910	773.7		19910	1800.0		19910	1800.1	NA	1800.0
462	15033	443.6		15033	1800.0		15033	1800.0	NA	1800.0
463	10522	2.4		10522	2.0		10522	38.7	10522	113.4
464	17236	3.7		17236	4.0		17236	59.9	17236	76.0
465	16187	6.3		16187	2.0		16187	91.7	16187	95.7
466	11634	0.5		11634	0.0		11634	39.2	11634	5.6
467	10914	0.4		10914	0.0		10914	33.7	10914	5.0
468	11024	0.4		11024	0.0		11024	6.9	11024	6.7
469	NA	1800.0		NA	1800.0		4087	1800.0	11522	1800.0
470	3979	317.5		NA	1800.0		3979	116.4	10300	1800.0
471	NA	1800.1		NA	1800.0		4219	1800.1	11031	1800.0
472	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
473	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
474	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
475	17189	50.0		17189	16.0		17189	784.0	17189	1195.9
476	11580	33.4		11580	18.0		11580	1800.0	11580	1401.5
477	9697	22.4		9697	28.0		9697	1800.1	9697	1027.4
478	15837	2.2		15837	0.0		15837	62.4	15837	29.5
479	16861	1.4		16861	0.0		16861	51.9	16861	19.2
480	12961	1.9		12961	0.0		12961	56.3	12961	33.7
481	NA	1800.0		NA	1800.0		3771	1461.3	12586	1800.0
482	3869	1800.0		NA	1800.0		3869	229.7	12110	1800.0
483	3419	645.7		NA	1800.0		3419	158.7	14192	1800.0
484	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
485	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
486	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
487	8094	32.1		8094	75.0		8094	1384.0	NA	1800.0
488	7818	31.5		7818	99.0		7818	1800.0	NA	1800.0
489	11857	125.2		11857	104.0		11857	1800.1	NA	1800.0
490	12343	2.5		12343	1.0		12343	72.6	12343	125.0
491	10120	3.3		10120	0.0		10120	59.0	10120	117.5
492	13695	5.4		13695	1.0		13695	128.3	13695	144.4
493	3482	1802.9		NA	1800.0		3482	575.8	9996	1800.0
494	3534	1854.2		NA	1800.0		3534	316.7	7894	1800.0
495	NA	1800.0		NA	1800.0		3708	1310.0	8961	1800.0
496	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
497	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
498	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
499	18371	1210.7		18371	447.0		NA	1800.0	NA	1800.0
500	11892	190.6		11892	286.0		NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
501	9333	203.5		9333	391.0		NA	1800.0	NA	1800.0
502	12505	10.9		12505	5.0		12505	494.7	12505	438.0
503	11695	11.0		11695	5.0		11695	226.9	11695	469.5
504	9471	5.7		9471	5.0		9471	158.8	9471	440.8
505	16348	169.0		NA	1800.0		16348	1610.4	NA	1800.0
506	NA	1800.0		NA	1800.0		27810	1800.0	NA	1800.0
507	22051	1514.6		NA	1800.0		22051	1800.1	NA	1800.0
508	19279	0.3		19279	3.0		19279	0.7	19279	3.0
509	15027	0.3		15027	1.0		15027	0.7	15027	1.3
510	14494	0.9		14494	2.0		14494	22.4	14494	3.9
511	18053	0.1		18053	0.0		18053	0.1	18053	0.1
512	13505	0.0		13505	0.0		13505	0.2	13505	0.0
513	16651	0.1		16651	0.0		16651	0.1	16651	0.1
514	18747	0.0		18747	0.0		18747	0.1	18747	0.0
515	17855	0.0		17855	0.0		17855	0.2	17855	0.0
516	26307	0.0		26307	0.0		26307	0.1	26307	0.0
517	NA	1800.0		NA	1800.0		15998	1800.1	NA	1800.0
518	NA	1800.0		NA	1800.0		21237	1800.0	NA	1800.0
519	NA	1800.0		NA	1800.0		25460	1800.0	NA	1800.0
520	17304	51.0		17304	477.0		17304	303.3	17304	1800.0
521	26405	48.6		26405	277.0		26405	231.8	NA	1800.0
522	20264	36.6		20264	264.0		20264	334.8	20264	1793.3
523	19128	0.9		19128	0.0		19128	26.3	19128	10.3
524	21163	2.8		21163	1.0		21163	56.8	21163	16.8
525	13952	1.5		13952	0.0		13952	40.0	13952	9.2
526	18035	0.2		18035	0.0		18035	4.1	18035	1.4
527	27193	0.2		27193	0.0		27193	3.5	27193	0.7
528	14341	0.4		14341	0.0		14341	4.1	14341	1.1
529	NA	1800.2		NA	1800.0		12804	1800.0	NA	1800.0
530	NA	1800.0		NA	1800.0		13902	1800.1	NA	1800.0
531	NA	1800.0		NA	1800.0		14048	1800.3	NA	1800.0
532	14178	926.2		14178	1800.0		14178	1800.0	NA	1800.0
533	NA	1800.0		NA	1800.0		20577	1800.1	NA	1800.0
534	14830	766.3		NA	1800.0		14830	1800.0	NA	1800.0
535	16045	9.2		16045	6.0		16045	414.1	16045	627.6
536	22226	13.3		22226	5.0		22226	112.9	22226	685.3
537	24899	22.6		24899	4.0		24899	383.6	24899	725.8
538	18448	1.4		18448	0.0		18448	84.1	18448	15.3
539	13831	0.9		13831	0.0		13831	15.1	13831	17.2
540	17335	1.7		17335	0.0		17335	79.0	17335	24.6
541	NA	1800.0		NA	1800.0		13289	1800.0	NA	1800.0
542	NA	1800.0		NA	1800.0		13114	1800.0	NA	1800.0
543	NA	1800.0		NA	1800.0		11007	1800.1	NA	1800.0
544	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
545	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
546	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
547	17443	110.7		17443	77.0		17443	1800.1	NA	1800.0
548	23058	144.1		23058	57.0		23058	1800.3	NA	1800.0
549	24689	150.2		24689	122.0		24689	1800.1	NA	1800.0
550	18276	3.2		18276	1.0		18276	109.0	18276	185.6
551	21556	5.9		21556	1.0		21556	185.0	21556	167.2
552	18300	3.5		18300	1.0		18300	108.4	18300	206.7
553	NA	1800.0		NA	1800.0		8763	1800.1	NA	1800.0
554	NA	1800.0		NA	1800.0		9868	1800.1	NA	1800.0
555	NA	1800.0		NA	1800.0		9142	1800.0	NA	1800.0
556	NA	1800.0		NA	1800.0		NA	1867.8	NA	1800.0
557	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
558	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
559	15460	242.4		15460	480.0		NA	1800.1	NA	1800.0
560	18239	375.5		18239	603.0		NA	1800.1	NA	1800.0
561	16964	595.7		16964	393.0		NA	1800.1	NA	1800.0
562	16636	19.9		16636	6.0		16636	1800.1	16636	1383.8
563	17875	20.7		17875	6.0		17875	1800.1	17875	1487.2
564	20584	26.5		20584	5.0		20584	739.6	20584	1402.8
565	NA	1800.0		NA	1800.0		8624	1800.1	NA	1800.0
566	NA	1800.0		NA	1800.0		7877	1800.1	NA	1800.0
567	NA	1800.7		NA	1800.0		6874	1800.1	NA	1800.0
568	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
569	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
570	NA	1800.0		NA	1800.0		NA	1953.9	NA	1800.0
571	NA	1800.0		17148	1800.0		NA	1800.0	NA	1800.0
572	NA	1800.3		NA	1800.0		NA	1800.0	NA	1800.0
573	NA	1800.0		23267	1800.0		23267	1801.9	NA	1800.0
574	25706	73.8		25706	22.0		25706	1804.4	NA	1800.0
575	21418	65.5		21418	19.0		21418	1803.1	NA	1800.0
576	15978	63.5		15978	25.0		15978	1800.1	NA	1800.0
577	NA	1800.0		NA	1800.0		7262	1800.0	NA	1800.0
578	NA	1800.0		NA	1800.0		6996	1800.1	NA	1800.0
579	NA	1800.0		NA	1800.0		7889	1800.1	NA	1800.0
580	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
581	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
582	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
583	NA	1800.0		NA	1800.0		NA	1800.4	NA	1800.0
584	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
585	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
586	19668	193.1		19668	68.0		NA	1887.6	NA	1800.0
587	17266	127.5		17266	66.0		NA	1800.2	NA	1800.0
588	18415	133.0		18415	65.0		18415	1800.2	NA	1800.0
589	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
590	NA	1800.3		NA	1800.0		NA	1800.0	NA	1800.0
591	NA	1800.0		NA	1800.0		23480	1800.0	NA	1800.0
592	25335	11.3		25335	32.0		25335	64.0	25335	183.5
593	18650	3.4		18650	19.0		18650	48.8	18650	392.7
594	21187	5.2		21187	8.0		21187	37.1	21187	69.9
595	20561	0.3		20561	0.0		20561	2.5	20561	1.2
596	36818	0.3		36818	0.0		36818	2.9	36818	2.7
597	24493	0.4		24493	0.0		24493	3.0	24493	1.8
598	18807	0.1		18807	0.0		18807	0.9	18807	0.2
599	21979	0.1		21979	0.0		21979	0.7	21979	0.2
600	19301	0.1		19301	0.0		19301	0.7	19301	0.2
601	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
602	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
603	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
604	21716	315.6		NA	1800.0		21716	1800.1	NA	1800.0
605	22611	794.0		NA	1800.0		22611	1800.1	NA	1800.0
606	26994	1220.1		NA	1800.0		26994	1800.0	NA	1800.0
607	25600	7.1		25600	4.0		25600	82.7	25600	492.6
608	26986	14.8		26986	6.0		26986	542.5	26986	366.2
609	21232	7.1		21232	6.0		21232	91.4	21232	386.5
610	32580	0.6		32580	0.0		32580	14.5	32580	11.4
611	22638	0.7		22638	0.0		22638	12.6	22638	16.5
612	23912	1.6		23912	0.0		23912	135.9	23912	19.8
613	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
614	NA	1800.0		NA	1800.0		NA	1802.4	NA	1800.0
615	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
616	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	Best UB	CPU	Best UB	Best UB	CPU	Best UB	CPU
617	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
618	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
619	43412	220.6		43412	96.0		43412	1800.1	NA	1800.0
620	37484	229.9		37484	43.0		NA	1800.0	NA	1800.0
621	26521	156.1		26521	118.0		26521	1800.1	NA	1800.0
622	24385	4.6		24385	1.0		24385	220.2	24385	262.2
623	27302	8.0		27302	1.0		27302	241.3	27302	450.1
624	24523	4.0		24523	3.0		24523	201.0	24523	279.0
625	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
626	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
627	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
628	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
629	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
630	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
631	21025	592.9		21025	1539.0		NA	1800.1	NA	1800.0
632	23439	805.1		NA	1800.0		NA	1800.3	NA	1800.0
633	20827	651.8		20827	1396.0		NA	1800.2	NA	1800.0
634	22175	27.7		22175	14.0		22175	1800.1	NA	1800.0
635	25993	47.7		25993	7.0		25993	1800.0	NA	1800.0
636	26290	39.2		26290	5.0		26290	1079.9	NA	1800.0
637	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
638	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
639	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
640	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
641	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
642	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
643	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
644	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
645	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
646	21954	128.3		21954	39.0		NA	1800.1	NA	1800.0
647	26932	171.2		26932	29.0		NA	1800.1	NA	1800.0
648	20096	101.2		20096	49.0		NA	1800.1	NA	1800.0
649	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
650	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
651	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
652	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
653	NA	1800.0		NA	1800.0		NA	1800.3	NA	1800.0
654	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
655	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
656	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
657	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
658	19780	270.3		19780	172.0		NA	1800.1	NA	1800.0
659	30129	563.9		30129	213.0		NA	1800.4	NA	1800.0
660	25610	358.5		25610	170.0		NA	1800.1	NA	1800.0
661	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
662	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
663	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
664	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
665	NA	1800.0		NA	1800.0		NA	1800.3	NA	1800.0
666	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
667	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
668	NA	1801.3		NA	1800.0		NA	1800.1	NA	1800.0
669	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
670	16142	377.3		16142	632.0		NA	1800.5	NA	1800.0
671	29475	1692.6		29475	665.0		NA	1800.2	NA	1800.0
672	23379	867.4		23379	846.0		NA	1800.3	NA	1800.0
673	NA	1801.2		NA	1800.0		NA	1800.0	NA	1800.0
674	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
675	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
676	35295	59.9		35295	45.0		35295	352.7	NA	1800.0
677	39047	33.8		39047	92.0		39047	172.7	NA	1800.0
678	32900	38.2		32900	34.0		32900	179.0	NA	1800.0
679	24369	0.9		24369	0.0		24369	13.0	24369	20.5
680	26485	1.3		26485	0.0		26485	158.2	26485	16.3
681	29781	1.6		29781	0.0		29781	267.4	29781	20.3
682	33628	0.2		33628	0.0		33628	5.8	33628	0.6
683	29090	0.3		29090	0.0		29090	7.0	29090	1.0
684	28024	0.4		28024	0.0		28024	8.7	28024	0.6
685	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
686	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
687	NA	1800.6		NA	1800.0		NA	1800.0	NA	1800.0
688	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
689	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
690	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
691	37208	31.1		37208	36.0		37208	310.2	NA	1800.0
692	29568	41.7		29568	21.0		29568	1800.1	NA	1800.0
693	33799	36.8		33799	22.0		33799	1800.0	NA	1800.0
694	27972	2.5		27972	0.0		27972	249.8	27972	88.3
695	33089	2.4		33089	1.0		33089	211.6	33089	64.2
696	35851	2.2		35851	1.0		35851	287.5	35851	93.2
697	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
698	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
699	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
700	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
701	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
702	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
703	23918	503.3		23918	791.0		NA	1800.1	NA	1800.0
704	30664	690.3		30664	565.0		NA	1800.1	NA	1800.0
705	24291	805.9		24291	683.0		24291	1800.6	NA	1800.0
706	34483	26.1		34483	10.0		34483	1800.1	34483	1800.0
707	37600	18.2		37600	7.0		NA	1800.1	37600	1800.0
708	36271	42.6		36271	7.0		NA	1800.1	NA	1800.0
709	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
710	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
711	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
712	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
713	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
714	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
715	NA	1800.0		26579	1800.0		NA	1800.1	NA	1800.0
716	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
717	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
718	37712	200.9		37712	76.0		NA	1800.2	NA	1800.0
719	29883	157.7		29883	61.0		NA	1800.2	NA	1800.0
720	25538	125.4		25538	39.0		NA	1800.4	NA	1800.0
721	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
722	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
723	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
724	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
725	NA	1800.0		NA	1800.0		NA	1801.5	NA	1800.0
726	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
727	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
728	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
729	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
730	33617	466.6		33617	139.0		NA	1800.5	NA	1800.0
731	29221	514.5		29221	288.0		NA	1800.2	NA	1800.0
732	40523	834.3		40523	313.0		NA	1801.0	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU	Best UB	CPU
733	NA	1800.4		NA	1800.0		NA	1800.1	NA	1800.0
734	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
735	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
736	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
737	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
738	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
739	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
740	NA	1800.0		NA	1800.0		NA	1800.4	NA	1800.0
741	NA	1800.0		NA	1800.0		NA	1801.0	NA	1800.0
742	24732	1123.2		24732	1087.0		NA	1800.1	NA	1800.0
743	NA	1800.0		31647	1471.0		NA	1800.1	NA	1800.0
744	23682	1504.5		23682	920.0		NA	1800.1	NA	1800.0
745	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
746	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
747	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
748	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
749	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
750	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
751	NA	1800.7		NA	1800.0		NA	1800.2	NA	1800.0
752	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
753	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
754	NA	1800.0		28535	1800.0		NA	1800.3	NA	1800.0
755	NA	1800.0		33402	1800.0		NA	1800.1	NA	1800.0
756	NA	1800.0		34707	1800.0		NA	1800.1	NA	1800.0
757	NA	1800.1		NA	1800.0		NA	1800.0	NA	1800.0
758	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
759	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
760	32705	197.6		32705	822.0		32705	688.3	NA	1800.0
761	26052	51.3		26052	585.0		26052	1645.0	NA	1800.0
762	25915	103.2		25915	781.0		25915	1800.1	NA	1800.0
763	26798	1.8		26798	4.0		26798	125.2	26798	89.1
764	21406	2.7		21406	2.0		21406	145.4	21406	70.6
765	36729	2.6		36729	1.0		36729	219.2	36729	107.3
766	25792	0.6		25792	0.0		25792	15.9	25792	1.8
767	33701	0.5		33701	0.0		33701	14.9	33701	3.2
768	36685	0.5		36685	0.0		36685	12.7	36685	2.7
769	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
770	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
771	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
772	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
773	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
774	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
775	33302	117.5		33302	46.0		33302	1800.1	NA	1800.0
776	37593	101.9		37593	125.0		37593	1800.1	NA	1800.0
777	31034	109.1		31034	38.0		31034	1800.0	NA	1800.0
778	44324	5.9		44324	1.0		44324	344.7	44324	344.8
779	38561	9.7		38561	1.0		38561	421.7	38561	427.3
780	40161	7.1		40161	1.0		40161	351.1	40161	388.1
781	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
782	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
783	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
784	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
785	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
786	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
787	NA	1800.0		31113	1177.0		NA	1800.1	NA	1800.0
788	NA	1800.0		40357	744.0		NA	1800.1	NA	1800.0
789	28306	1226.8		28306	990.0		NA	1800.1	NA	1800.0
790	38115	71.1		38115	15.0		NA	1800.2	NA	1800.0

Table B.1. Detailed experimental results (cont.).

	BBA		BBA w/o Inf.	Diving		Weak - single	Weak - multi		BBSS	
Ins.	Best UB	CPU	Best UB CPU	Best UB	CPU	Best UB CPU	Best UB	CPU	Best UB	CPU
791	28566	65.5		28566	11.0		NA	1800.3	NA	1800.0
792	34622	50.3		34622	18.0		NA	1800.1	NA	1800.0
793	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
794	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
795	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
796	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
797	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
798	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
799	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
800	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
801	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
802	29694	240.3		29694	156.0		NA	1800.3	NA	1800.0
803	25143	167.6		25143	124.0		NA	1800.2	NA	1800.0
804	38372	263.2		38372	176.0		NA	1800.1	NA	1800.0
805	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
806	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
807	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
808	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
809	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
810	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
811	NA	1800.0		NA	1800.0		NA	1800.4	NA	1800.0
812	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
813	NA	1800.0		NA	1800.0		NA	1800.6	NA	1800.0
814	NA	1800.0		40486	476.0		NA	1800.1	NA	1800.0
815	27711	1356.5		27711	614.0		NA	1800.1	NA	1800.0
816	NA	1800.0		39745	542.0		NA	1800.1	NA	1800.0
817	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
818	NA	1800.0		NA	1800.0		NA	1800.0	NA	1800.0
819	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
820	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
821	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
822	NA	1821.3		NA	1800.0		NA	1800.1	NA	1800.0
823	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
824	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
825	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
826	NA	1800.0		34783	1800.0		NA	1800.1	NA	1800.0
827	NA	1800.0		35724	1800.0		NA	1800.1	NA	1800.0
828	NA	1800.0		42560	1800.0		NA	1800.3	NA	1800.0
829	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
830	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
831	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
832	NA	1800.0		NA	1800.0		NA	1801.7	NA	1800.0
833	NA	1800.0		NA	1800.0		NA	1800.6	NA	1800.0
834	NA	1800.0		NA	1800.0		NA	1800.3	NA	1800.0
835	NA	1800.0		NA	1800.0		NA	1800.2	NA	1800.0
836	NA	1800.0		NA	1800.0		NA	1800.5	NA	1800.0
837	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
838	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0
839	NA	1800.0		33736	1800.0		NA	1801.5	NA	1800.0
840	NA	1800.0		NA	1800.0		NA	1800.1	NA	1800.0